Open Source Jahrbuch 2005

Zwischen Softwareentwicklung und Gesellschaftsmodell

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

Open Source Jahrbuch 2005

Zwischen Softwareentwicklung und Gesellschaftsmodell

Herausgegeben und bearbeitet von:

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

Clemens Brandt
Student der Informatik, TU Berlin

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

Annika Held
Studentin der Medienberatung, TU Berlin

Katja Luther
Studentin der Informatik, TU Berlin

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

Wolfram Riedel
Student der Kommunikationswissenschaft
und der Informatik, TU Berlin

Patrick Stewin
Student der Informatik, TU Berlin

Sebastian Ziebell
Student der Informatik, TU Berlin

Bastian Zimmermann
Student der Informatik, TU Berlin

2005

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 2005
Zwischen Softwareentwicklung und Gesellschaftsmodell
Bärwolff, Matthias; Gehring, Robert A.; Lutterbeck, Bernd (Hrsg.)

Berlin, 2005 – Lehmanns Media – LOB.de
ISBN: 3–86541–059–6

Satz und Layout: Wolfram Riedel, Berlin
unter Verwendung des Textsatzsystems LaTeX
Einbandgestaltung: Matthias Bärwolff, Berlin
unter Verwendung des Textes der *GNU General Public License*
Druck und Verarbeitung: TRIGGERagent, Berlin

Printed in the European Union.

Inhalt

Kapitel 3 – Ökonomie

Kapitel 4 – Recht und Politik

Kapitel 5 – Gesellschaft

Kapitel 6 – Open Content

Kapitel 7 – Open Innovations

Vorwort der Herausgeber

Matthias Bärwolff, Robert A. Gehring
und Bernd Lutterbeck

Als wir auf der CeBIT im März letzten Jahres das Open Source Jahrbuch 2004 der Öffentlichkeit vorstellten, wussten wir natürlich nicht, ob wir unsere Ziele erreichen würden. Heute wissen wir mehr. Die nüchternen Zahlen sprechen dafür, dass das Buch keinen schlechten Start gehabt hat: Ein Jahr später haben wir das Buch mehr als 500-mal verkauft und verzeichnen 20 000 Downloads der PDF-Version von unserem Server. Durch die Erlöse war es uns möglich, das neue Jahrbuch 2005 mit nur geringen Zuschüssen zu produzieren.

Unser Ziel haben wir also erreicht: ein umfassendes Kompendium zum Thema Open Source für eine deutschsprachige Leserschaft vorzulegen, mit dem wir Verantwortlichen in Politik und Wirtschaft eine Entscheidungshilfe in die Hand geben wollten, die praktisch sämtliche Aspekte von Open Source abdeckt. Von Fallbeispielen über Migrationen zu Open Source bis hin zu gesellschaftlichen Aspekten waren alle relevanten Themen im Buch vertreten. Zugleich schaffte das Buch die nicht immer leichte Gratwanderung zwischen Praxisrelevanz und dem akademischen Anspruch, wissenschaftliche Beiträge zu hochaktuellen Forschungsthemen bereitzustellen.

Die Kritiken zum Open Source Jahrbuch 2004 waren mehrheitlich positiv. Gerade in den ersten Tagen nach der Veröffentlichung erreichten uns viele E-Mails von Lesern mit Lob und Tadel, aber auch mit Anregungen und Ideen für das nächste Jahrbuch. Einige dieser Anregungen haben wir im vorliegenden Buch aufgegriffen.

Den pragmatischen Ansatz der Strukturierung nach Themengebieten haben wir angesichts der Vielzahl der Stimmen in der aktuellen Diskussion um Open Source beibehalten. Neben den gewohnten Kapiteln „Technik“, „Recht und Politik“, „Ökonomie“, „Gesellschaft“ und „Fallbeispiele“ finden sich in diesem Jahrbuch die neu hinzugekommenen Kapitel „Open Innovations“ und „Open Content“, die sich den beiden wesentlichen neuen Trends des letzten Jahres widmen.

Auf Anregung eines Besuchers an unserem CeBIT-Stand im letzten Jahr findet sich im aktuellen Jahrbuch ein Stückchen echter Open-Source-Quellcode – zum Anfassen, sozusagen. Nachdem die Redaktion einen Wettbewerb um das beste Open-Source–Programm mit höchstens 500 Zeilen Quellcode ausgelobt hatte, wählten die Besucher

unserer Webseite aus allen Einsendungen den Sieger, dessen Programm wir hier abgedruckt haben.[1]

In der Arbeit an diesem Jahrbuch haben wir darauf geachtet, Herausgeber und Redaktion stärker als im Vorjahr voneinander zu trennen. Die Unabhängigkeit der Redaktion soll sicherstellen, dass das Jahrbuch ein Projekt von Studierenden der TU Berlin bleibt. Die Redaktion sollte die durchaus hohen Ansprüche der Herausgeber umsetzen, auch wenn dies nicht immer einfach war. Ihre eigenen hohen Ansprüche, von der Qualität der Artikel bis hin zur Produktion des Buches selbst, mussten sich besonders bewähren, etwa wenn es galt, Beiträge abzulehnen, die nicht den Anforderungen der Redakteure entsprachen. Natürlich gab es kontroverse Diskussionen über die Aufnahme mancher Beiträge, in denen nicht immer einstimmige Entscheidungen getroffen wurden. So haben wir auch Artikel ins Buch aufgenommen, deren Wissenschaftlichkeit und Grundaussage ein Teil der Herausgeber und Redakteure ernsthaft in Zweifel zog und für falsch erachtete.

Sehr erfreulich war für uns, dass praktisch alle Autoren der Bitte nachgekommen sind, ihre Beiträge unter eine CC-Lizenz zu stellen – ganz im Sinne von Open Source. Auch der Beitrag von Walter Seemayer und Jason Matusow von Microsoft im Kapitel „Ökonomie" macht hier keine Ausnahme.

Bei der Arbeit am Jahrbuch haben wir deutlich, an der Resonanz der Autoren, wie auch am Enthusiasmus der Redaktion gespürt, dass das Thema Open Source nichts von seiner Relevanz und Aktualität verloren hat. Beiträge aus der Praxis, wie der über das mit dem „Open Source Best Practice Award 2004"[2] ausgezeichnete Projekt „GENOMatch" der Tembit Software GmbH,[3] bis hin zu aktuellen wissenschaftlichen Beiträgen aus der amerikanischen Innovationsforschung[4] belegen dies eindrucksvoll.

Hervorheben möchten wir an dieser Stelle das beispielhafte Engagement, mit dem Wolfram Riedel der Redaktion den Satz des Buches in LaTeX ermöglicht hat. Herzlich bedanken möchten wir uns auch bei den Autoren, die uns ihre Beiträge, wie schon im letzten Jahr, unentgeltlich zur Verfügung gestellt haben und der Buchhandlung Lehmanns, die uns beim Vertrieb des Jahrbuchs unterstützt.

Wir hoffen, dass uns mit dem vorliegenden Buch gelungen ist, was wir uns vorgenommen haben: Ein aktuelles und qualitativ hochwertiges Kompendium zum Thema Open Source zu schaffen, das eine breite Leserschaft in Wissenschaft und Praxis anspricht und die Diskussion um Open Source in Deutschland nachhaltig bereichert.

Januar 2005
Bernd Lutterbeck
Robert A. Gehring
Matthias Bärwolff

1 Der Beitrag von Oliver Feiler findet sich im Kapitel „Technik" in diesem Band.
2 Der Preis wurde verliehen u. a. von der Lightwerk GmbH in Kooperation mit dem Fraunhofer IAO und dem Linux-Verband.
3 Siehe den Beitrag von Broder Schümann und Dennis Petrov im Kapitel „Fallbeispiele".
4 Siehe den Beitrag von Eric von Hippel im Kapitel „Open Innovations".

Vorwort des Open Source Jahrbuchs 2004

ROBERT A. GEHRING UND BERND LUTTERBECK

Es ist schon erstaunlich, in welch kurzer Zeit sich das Phänomen „Open Source" weltweit ausgebreitet hat – eine ohne das Internet undenkbare Entwicklung. Man kann dies gut an einigen Arbeiten nachvollziehen, die in unserer Forschungsgruppe im Abstand von mehreren Jahren geschrieben wurden: Gehring (1996), Ardal (2000) und Leiteritz (2002). Während Gehring (1996) wohl weltweit die erste Arbeit war, die sich intensiv und grundlegend mit rechtlichen Aspekten auseinander setzte, verfolgten die beiden späteren Arbeiten eine ökonomische Zielrichtung.

Atila Ardal hat damals (2000) Neuland betreten. Es gab praktisch noch keine Literatur, die bei der Beantwortung der von ihm aufgeworfenen Fragen (im Kern: Ist Open Source betriebswirtschaftlich sinnvoll?) geholfen hätte. Er musste aus diesem Grunde noch viel mit Analogien arbeiten. Mehr als Anfänge eines theoretischen Modells konnte man da noch nicht erwarten.

Zwei Jahre später (2002) hat sich für Raphael Leiteritz die Situation schon entscheidend verändert. Zum einen waren sehr präzise Aussagen über die betriebliche Praxis von Open Source möglich: Raphael Leiteritz hatte als Gründer und späterer CEO eines über einige Jahre sehr aktiven Start-up-Unternehmens viele positive und negative Erfahrungen bei der Vermarktung von Open-Source-Software sammeln können.

Hinzu kam, dass das neue Phänomen inzwischen fachübergreifend zum Gegenstand intensiver Forschungen geworden war. Im Ergebnis lagen mittlerweile zahlreiche, teils hochwertige Beiträge aus den unterschiedlichsten Wissenschaftsdisziplinen vor – Ökonomie, Soziologie, Recht, Informatik –, die zu zahlreichen Aspekten von Open Source Stellung bezogen. Daraus wurde klar, dass es sich bei Open Source weder um eine Eintagsfliege noch um einen auf Software beschränkten Ansatz handelte.[5] Um nicht unterzugehen in dieser Vielfalt, hat Raphael Leiteritz im Rahmen seiner Diplomarbeit ein heuristisches Modell entwickelt, mit dem er alle Aspekte einigermaßen vernünftig ordnen konnte. Wir finden, es leistet auch heute noch gute Dienste bei der Orientierung (siehe Abbildung 1).

Natürlich kann kein Wissenschaftler allein mehr diese durchaus beeindruckende Vielfalt in all ihren Facetten überblicken.[6] Von der Open-Source-Bewegung kann

5 Das ist auch der Grund, warum wir vorrangig von Open Source sprechen: Die Bedeutung des Open–Source-Ansatzes reicht unserer Meinung nach, wiewohl aus der Software-Entwicklung hervorgegangen, weiter. Der offene Umgang mit Wissen betrifft eben mehr als nur Software. Dort wo wir spezifische Softwarefragen ansprechen, wird, wo notwendig, zwischen Freier Software und Open-Source-Software differenziert.

6 Einen guten Überblick verschafft die Open-Source-Bibliographie von O'Reilly (2002).

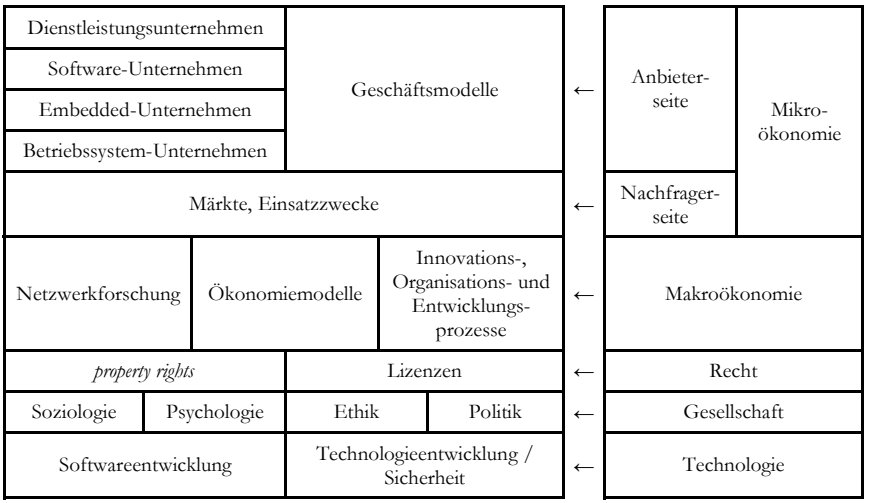


Abbildung 1: Heuristisches Modell nach Leiteritz (2002)

man aber lernen, dass man die eigenen Schwächen durch neue, andere Formen der Kooperation überwinden kann. Schon seit einigen Jahren ist deshalb Open Source fester Bestandteil unseres Lehrplanes. Das gilt sowohl für die Behandlung des Themas als auch für die Verwendung von Elementen der Open-Source-Methode in der Lehre. Ein Ergebnis dessen ist dieses Buch.

Entstanden ist die Idee zu diesem Buch vor fast drei Jahren, als die (nunmehrigen) Herausgeber diskutierten, wie man Studenten das Thema *praktisch* nahe bringen könnte. Jeweils im Sommersemester eines Jahres führen wir Praktika durch, in deren Rahmen sich unsere Studierenden mit einer Vielfalt von Problemen aus dem Bereich Informatik und Gesellschaft auseinander setzen. Ermutigt durch den überwältigenden Erfolg der WIDI-Studie[7] des Jahres 2000, einer empirischen Studie über Free- bzw. Open-Source-Softwareentwickler, beschlossen wir, ein so ambitioniertes Unternehmen in Angriff zu nehmen, wie es ein Buch ist. Orientierung bot dabei die lange Tradition an vielen Hochschulen in den USA, renommierte Fachzeitschriften durch eine Redaktion aus Studierenden betreuen zu lassen. In einer Zeit, in der Universitäten und Hochschulen finanziell ausbluten und die Kosten für Fachliteratur selbst für Hochschullehrer in unerschwingliche Höhen klettern, muss man neue Wege gehen – Open-Source-Wege. Dabei kann man, das zeigt unsere Erfahrung, auf die Unterstützung vieler unterschiedlicher Menschen bauen.

An dieser prominenten Stelle möchten wir vor allem den „Machern" dieses Jahrbuchs unsere Anerkennung aussprechen – der Redaktion. Alle sind sie ja Studierende, die fast ein ganzes Jahr lang einen Teil ihrer Zeit für das Gelingen des Projekts ge-

7 Robles, Scheider, Tretkowski und Weber (2000).

opfert haben. Alles, was man durch irgendwelche Uni-Scheine abgelten kann, war längst geschehen. Geblieben ist ein freiwilliger Einsatz für eine sinnvolle Sache. Man weiß inzwischen sehr gut, dass intrinsische Motivation ein wesentlicher, vielleicht der wichtigste Teil des Geheimnisses von Open-Source-Projekten ist. Dieses Buch mag ein weiterer Beweis dafür sein. Profitiert hat die Gruppe ganz sicher auch vom Einsatz von Svetlana Kharitoniouk, der einzigen Frau in der Gruppe, die sich zutraute, ein solches Unternehmen zu leiten. Ihre ökonomischen Kenntnisse waren dabei erkenntlich eine gute Voraussetzung.

Bei der Auswahl der Beiträge hat die Redaktion sich – haben wir Herausgeber uns – um hohe Qualität bemüht. Das Vorschlagsrecht lag in erster Linie bei der Redaktion, die Herausgeber sind herbeigeeilt, wenn ihre Hilfe nötig war. Das Ergebnis kann sich, wie wir finden, sehen lassen. Natürlich kann man sich noch verbessern – eine Aufgabe für das nächste Jahrbuch, 2005. Die Vorbereitungen dazu haben bereits begonnen.

Wir haben Wert darauf gelegt, dass verschiedene Autorinnen und Autoren zu ähnlichen Themen zu Wort kommen. Die Vielfalt der Meinungen und das ganze Spektrum der Thesen zu erfassen war unser erstes Ziel. Das ist uns teilweise gelungen. Nicht immer hatten die Wunschautoren Zeit, und nicht immer wurden die Anfragen der Studierenden ernst genommen. So kommt es, dass in einigen Rubriken erkennbar ein Ungleichgewicht entstanden ist.

Man sollte nicht verschweigen, dass es auch Konflikte gab. Wir haben z. B. einen Anwenderbericht einwerben können, der sich sehr negativ zum Einsatz von Open–Source-Software in dem entsprechenden Bereich äußert – darf man so etwas abdrucken? Das Redaktionsteam hat intensiv gerade über diesen Fall diskutiert. Das Ergebnis: Man darf nicht nur, man muss sogar! Was wäre damit gewonnen, den Lesern und Leserinnen eine heile Welt vorzuspielen? Und ist nicht gerade eine Einsicht aus der Open-Source-Praxis, Fehler oder auch nur Schwierigkeiten offen zu legen, zu diskutieren – und es beim nächsten Mal besser zu machen?

Leider ist es uns nicht gelungen, diesen prinzipiellen Gedanken der Fairness dem Unternehmen Microsoft verständlich zu machen. Auch nach zahlreichen mündlichen wie schriftlichen Anfragen auf unterschiedlichen Hierarchiestufen hat sich das Unternehmen nicht darauf einlassen können, mit einem eigenen Beitrag seine Position zu erläutern – bedauerlich. Letztendlich nachvollziehen können wir diese Haltung nicht. Verfolgt man die aktuellen Aktivitäten des Unternehmens, so hat es den Anschein, als hätte sich Microsoft entschlossen, der anschwellenden Diskussion mit Werbekampagnen und bestellten Gutachten (z. B. Kooths, Langenfurth und Kalwey 2003) zu begegnen. Wir denken, dass eine große Anzahl der Beiträge dieses Jahrbuchs den Nachweis erbringt, dass die Diskussion weiter zu fassen ist, als es sich den Autoren dieses Gutachtens schon erschlossen hat.

Unser großer Dank gilt den Autorinnen und Autoren, die das manchmal penetrante Nachbohren geduldig, vielleicht auch zähneknirschend über sich haben ergehen lassen. Wenn nicht alle auf ein Honorar verzichtet hätten, gäbe es dieses Buch nicht, nicht zu diesem Preis, und es gäbe auch nicht die zugehörige Webseite, auf der die Beiträge kostenlos zugänglich gemacht werden.

Die Kooperation mit der Buchhandlung Lehmann ermöglicht es uns, selbst ohne

einen Verlag im Rücken das technische Procedere von Druck und Vertrieb mit vertretbarem Aufwand abzuwickeln; auch das finanzielle Risiko ruht dergestalt nicht allein auf unseren Schultern. Der Verzicht auf den Markennamen eines renommierten Verlages gestattet es zudem, dass die Rechte an den Texten bei den Autoren verbleiben – auch das ist ein Stück mehr Open Source.

Wen wünschen wir uns als Leser?

In erster Linie natürlich all jene, die schon selbst in einem Open-Source-Prozess stecken, sei es als Softwareentwickler, Unternehmer, Wissenschaftler oder Studierender. Ihnen wollen wir ein anschauliches Koordinatensystem an die Hand geben, ihnen zeigen, dass sie Teil eines Experimentes sind, dessen Größe zu erkennen das Alltagsgeschäft nicht immer gestattet.

Aus zahlreichen Gesprächen mit Berliner Politikern und Mitarbeitern aus Ministerien wissen wir auch, dass dort ein Bedarf an verlässlicher Information und Orientierung im Chaos besteht – vor allem bei denjenigen, die über den „Einkauf" und den Einsatz von Software und Systemen entscheiden müssen. Dieses Jahrbuch soll deshalb – auch anhand von Praktikern aus Wirtschaft und Verwaltung – einen kompakten Überblick über die intellektuelle und technologische Vielfalt des Phänomens Open Source bieten.

Und zu guter Letzt möchten wir den Skeptikern der Open-Source-Methode eine Anlaufstelle zur Prüfung ihrer Positionen geben. Auch ihre substantiierten Argumente sind zu berücksichtigen.

Open Source lebt vom „free flow of information". Vielleicht ist ja der offene „marketplace of ideas", das „knowledge commons", ein besseres Paradigma für den Aufbau der Informationsgesellschaft als der eingezäunte „marketplace of proprietary information goods"? Dieses Buch wird die Frage nicht beantworten, aber es zeigt ihre Berechtigung.

Berlin, im Januar 2004
Robert A. Gehring
Bernd Lutterbeck

Literatur

Ardal, A. (2000), Open Source – das Beispiel Linux: Ökonomische Analyse und Entwicklungsmodell eines erfolgreichen Betriebssystems, Diplomarbeit, Fachbereich Informatik der TU Berlin. http://ig.cs.tu-berlin.de/forschung/OpenSource/.

Gehring, R. A. (1996), 'Freeware, Shareware und Public Domain', Studienarbeit am Fachbereich Informatik der TU Berlin. http://ig.cs.tu-berlin.de/forschung/OpenSource/.

Kooths, S., Langenfurth, M. und Kalwey, N. (2003), Open-Source-Software. Eine volkswirtschaftliche Bewertung, Economic Research Studies 4, Muenster Institute for Computational Economics, Münster. http://mice.uni-muenster.de.

Leiteritz, R. (2002), Der kommerzielle Einsatz von Open-Source-Software und kommerzielle Open-Source-Geschäftsmodelle. Zur Bedeutung von Open-Source-Software in Unternehmen und als Grundlage für Geschäftsmodelle, Diplomarbeit, Fachbereich Informatik der TU Berlin. http://ig.cs.tu-berlin.de/forschung/OpenSource/.

O'Reilly (2002), *Open Source Bibliography*, 3. Aufl., O'Reilly & Associates Inc., Sebastopol.

Robles, G., Scheider, H., Tretkowski, I. und Weber, N. (2000), WIDI: Who Is Doing It? A research on Libre Software developers, Forschungsbericht, Fachbereich Informatik der TU Berlin. http://ig.cs.tu-berlin.de/forschung/OpenSource/.

Synopsis

Das Open Source Jahrbuch ist als umfassendes Nachschlagewerk gedacht. Dies bedeutet einerseits, dass das Jahrbuch das Thema Open Source in einem Umfang behandelt, der im deutschsprachigen Raum einzigartig sein dürfte. Andererseits heißt dies auch, dass sich nicht jeder Leser für alle Artikel des Jahrbuches interessieren wird. Folgender Überblick soll dazu dienen, von den verschiedenen Kapiteln bezüglich Inhalt und Herangehensweise einen ersten Eindruck zu vermitteln, ohne auf jeden Artikel speziell einzugehen. Detailliertere Einführungen zu den einzelnen Artikeln finden sich in den jeweiligen Kapiteleinleitungen.

Fallbeispiele

Für einen anschaulichen Einstieg in das Jahrbuch wurde ein Kapitel gewählt, das fast ausschließlich aus Erfahrungsberichten besteht. Diese erlauben eine konkrete Vorstellung von der Einsetzbarkeit von Open-Source-Software (OSS) in der Praxis. Neben aktuellen Strategien werden Beispiele aus der öffentlichen Verwaltung und der Wirtschaft gezeigt. Auffällig dabei ist der geschäftskritische Einsatz von u. a. zentral administrierbaren Applikationen, die eine überschaubare und somit leicht zu pflegende Architektur von verteilten Systemen erlauben. Die Beiträge zeigen, wie gerade durch die Nutzung von OSS (auch in Ergänzung zu proprietärer Software) dieser Faktor umgesetzt bzw. sogar verstärkt wird und somit deutliche wirtschaftliche Vorteile neben der Einsparung von Lizenzgebühren ermöglichen kann.

Technik

An die technischen Aspekte des vorangegangenen Kapitels anknüpfend, widmet sich das Kapitel „Technik" dem Ursprungsgebiet der Open-Source-Bewegung. Unter den vielen Entwicklungen in diesem Bereich, stechen vor allem die jüngsten Erfolge von OSS im Desktop-Bereich und die zunehmende Bedeutung von eingebetteten Systemen heraus. Aus diesem Grund behandelt das Kapitel „Technik" zum einen die Benutzbarkeit von OSS und zum anderen am Beispiel von „Embedded Linux" die Rolle, die OSS bei der Entwicklung eingebetteter Systeme, wie z. B. in Mobiltelefonen, spielt. Darüber hinaus befasst sich der Beitrag über Software-Evolution mit den erfolgreichen Methoden der offenen Softwareentwicklung. Der Abdruck des Quellcodes von „Snowbox", einem POP3-Server in Perl, soll gerade fachfremden Lesern einmal den Blick in ein „waschechtes" Open-Source-Programm ermöglichen.

Ökonomie

Das Kapitel „Ökonomie" vereint Artikel mit unterschiedlichen Zielsetzungen, gemein ist jedoch allen, dass sie für eine breite Leserschaft gedacht sind. Während Jens Mundhenke u. a. wichtiges ökonomisches Grundlagenwissen im Zusammenhang mit Software vermittelt, gehen die folgenden beiden Beiträge zwei konkreten Fragen nach, die im Rahmen von OSS immer wieder auftauchen. Der erste widmet sich der aktuellen Diskussion um das Spannungsfeld von offener Software und offenen Standards, während der zweite die aktuellen Erkenntnisse aus der empirischen Forschung über die Motivation von OSS-Entwicklern vorstellt. Daran anknüpfend vermitteln zwei Mitarbeiter aus Managementebene der Firma Microsoft die strategischen Erwägungen des Unternehmens hinter dessen „Shared-Source-Initiative". Der abschließende Artikel diskutiert den institutionellen Wandel duch Open Source und stellt die These auf, dass Open Source als „Meta-Idee" maßgeblich einen strukturellen Wandel in der Softwareindustrie vorantreibt.

Recht und Politik

Mit seinen zwei thematischen Schwerpunkten konzentriert sich das Kapitel auf rechtliche und politische Brennpunkte des letzten Jahres: zum einen Softwarepatente und ihre Auswirkung auf die Open-Source-Entwicklung und zum anderen die Rechtssicherheit von Open-Source-Lizenzen. Während die ersten beiden Artikel aktuelle Entwicklungen rund um Softwarepatente – insbesondere die Rolle der EU bei der Schaffung rechtlicher Rahmenbedingungen – thematisieren, beschäftigt sich Thomas Ebinger intensiv mit der ersten Entscheidung eines deutschen Gerichts zur *GNU General Public License* (GPL). Die aus Amerika stammende GPL ist weit verbreitet und die wohl bekannteste Lizenz für freie Software, ihre Gültigkeit in Deutschland war jedoch bis zum Urteil des Münchner Landgerichts vom 19. Mai 2004 rechtlich weitgehend ungeklärt.

Gesellschaft

Das Kapitel „Gesellschaft" zeigt, dass der Open-Source-Ansatz nicht auf Softwareentwicklung beschränkt bleibt, sondern sich deutlich darüber hinaus ausweitet. Es werden hierfür weniger technische und anwendungsbezogene Details begutachtet, als vielmehr die kulturellen und sozialen Aspekte von Open Source aus der Perspektive der Geisteswissenschaften betrachtet. Hierbei wird versucht, die Vielschichtigkeit von Open Source zu erfassen und der Bedeutung für die Gesellschaft nachzugehen. So widmet sich ein Artikel in einem geschichtlichen Rückblick den Motiven, die zur Freien Software Bewegung führten, während der nachfolgende Beitrag die umstrittene These vertritt, dass mit Freie Software die Tendenz einer neuen Gesellschaftsform vorhanden ist. Ein weiterer Beitrag orientiert sich an klassischen Philosophen und Soziologen, um Parallelen zwischen Open Source und gesellschaftlichen Vorgängen zu zeigen. Der abschließende Beitrag befasst sich mit der Frage, wie das Internet zukünf-

tig gestaltet sein sollte, um sowohl das Kreativitäts- als auch das Innovationspotential zu erhalten.

Open Content

Das Kapitel „Open Content" beleuchtet den Umgang mit der künstlerischen Seite des „geistigen Eigentums". Es wird dabei untersucht, ob das Open-Source-Modell auch auf Inhalte, sei es in Form von Texten oder Musik, anwendbar ist und inwieweit alternative Lizenzmodelle den Umgang mit diesen Inhalten beeinflussen. Zudem wird anhand der Fallbeispiele „Thinner Netlabel" und „Wikipedia" gezeigt, wie durch Kooperation und Offenheit Inhalte entstehen und welche Relevanz sie für die Gesellschaft haben.

Open Innovation

Das Kapitel zu „Open Innovation" führt drei Artikel zusammen, die das Spannungsfeld zwischen Innovation, Distribution und „geistigem Eigentum" ausloten. Alle Autoren, Bessen und Maskin, Wurch sowie von Hippel, zeigen anhand unterschiedlicher Fallbeispiele (von Gitarristen über Wissenschaftler bis hin zu Windsurfern), wie neues Wissen außerhalb des starren Rahmens einer bloß dogmatischen Vorstellung von „geistigem Eigentum" entsteht, wie traditionelle Institutionen an Bedeutung verlieren und neue entstehen. Gemeinsam ist den Beiträgen nicht nur der methodische Ansatz, sondern auch das Fazit: Der historische Kompromiss „geistiges Eigentum" muss nach Kosten und Nutzen neu bewertet werden. Und nur dort, wo der Nutzen die Kosten übersteigt, behält er weiterhin seine Daseinsberechtigung; wo das nicht der Fall ist, muss ein neuer Kompromiss ausgehandelt werden. Die Praxis außerhalb der juristischen Lehrbücher ist schon längst auf dem Weg dahin.

Kapitel 1

Fallbeispiele

„Certainly we think of it [Linux] as a competitor in the student and *hobbyist market*. But I really don't think in the commercial market we'll see it in any significant way."

– Bill Gates (1999) in „Yeah, That's the Ticket. . . ", Forbes.com Inc. 2004

Open-Source-Software in geschäftskritischen Einsatzgebieten

PATRICK STEWIN

1. Einleitung zum Kapitel „Fallbeispiele“

Dieses Kapitel soll dazu dienen, dem Leser einen umfassenden Überblick über die Einsatzgebiete von Open-Source-Software und Freier Software zu vermitteln.[1] Im Mittelpunkt stehen dabei große Institutionen wie Unternehmen und Behörden, welche oftmals eine „Vorreiterrolle“ einnehmen, bzw. die von ihnen durchgeführten Migrations- und Entwicklungsprojekte. Die hier zusammengestellten Artikel zeigen Ausgangslagen, überwundene Probleme und die Ergebnisse der aktuellen Projekte im letzten Jahr.

Es sei an dieser Stelle darauf hingewiesen, dass sämtliche Artikel der Autoren im Jahr 2004 verfasst wurden. Die Projekte sind mittlerweile fortgeschritten. Um Verwirrungen bzgl. der zeitlichen Angaben in den Berichten vorzubeugen, wurde nach Möglichkeit der zeitliche Stand an entsprechenden Stellen angegeben. Die folgenden Ausführungen führen thematisch an die Projektberichte heran.

2. Interoperabilität als Leitmotiv

IT-Infrastrukturen können auf zwei verschiedene Arten aufgebaut sein. Eine Möglichkeit ist der Aufbau eines homogenen Systems. Dort sind Produkte mit Technologien und Schnittstellen *eines* Herstellers zu finden, die nicht offen sein müssen, d. h. proprietär sind, da sie in der Regel nicht mit Produkten anderer Hersteller zusammenarbeiten müssen. Im Gegensatz dazu existieren heterogene Systeme.[2] Solche Systeme sind durch Komponenten und Produkte von unterschiedlichen Herstellern geprägt, die mit Hilfe von standardisierten Schnittstellen zusammenarbeiten bzw. kommunizieren müssen. Die Grundlage dieser Zusammenarbeit beschreibt die Fähigkeit zur Interoperabilität. Voß et al. (2004) definieren diesen Begriff unter anderem wie folgt:

1 Detaillierte Informationen, insbesondere zu den Unterschieden der Softwaremodelle Freie Software und Open-Source-Software, finden sich in Kharitoniouk und Stewin (2004, S. 2 ff.).

2 Zum Beispiel ist das Internet ein heterogenes System/Netzwerk.

> „Interoperabilität ist die Fähigkeit unabhängiger, heterogener Systeme möglichst nahtlos zusammen zu arbeiten, um Informationen auf effiziente und verwertbare Art und Weise auszutauschen bzw. dem Benutzer zur Verfügung zu stellen, ohne dass dazu gesonderte Absprachen zwischen den Systemen notwendig sind."

Bei der Entwicklung der IT-Infrastrukturen von Unternehmen, aber auch Behörden haben sich die Systeme im Laufe der Zeit unterschiedlich ausgeprägt. Bei manchen Institutionen haben sich kommerzielle Produkte etabliert, die in heterogenen Umgebungen über offene Standards bzw. Schnittstellen kommunizieren, also zu einem bestimmten Grad interoperabel sind. Bei anderen hingegen konnten sich auch Produkte mit proprietären Schnittstellen etablieren – diese stammen von einem Hersteller und können neben dem Vorteil, alles aus einer Hand zu beziehen auch Nachteile[3] mit sich bringen. In den letzten Jahren konnte sich auch Open-Source-Software (OSS), wie GNU/Linux in die verteilten Systeme von Institutionen einbringen (vgl. Wheeler 2005). Diese Art von Software basiert prinzipiell auf offenen Standards bzw. Schnittstellen und kann sich somit in entsprechende Systeme ohne Probleme integrieren.

Welchen Stellenwert der Begriff „Interoperabilität" für IT-Systeme großer Institutionen, wie beispielsweise Bundesbehörden hat, zeigt unter anderem die Software–Strategie der Bundesverwaltung (KBSt 2003*b*):

> „Die Bundesverwaltung verfolgt eine Softwarestrategie, die den Prinzipien der Interoperabilität und der offenen Standards verpflichtet ist."

Dazu fördert die Bundesverwaltung bzw. die Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) im Bundesministerium des Innern (BMI) auch Open-Source-Software. In KBSt (2004) heißt es:

> „[. . .] befasst sich die Projektgruppe Softwarestrategien vorrangig mit dem Thema Open-Source-Software. In Zusammenarbeit mit dem Bundesamt für Sicherheit in der Informationstechnik (BSI) werden derzeit mehrere Projekte vorangetrieben, die sich unter folgenden Stichworten zusammenfassen lassen:
>
> – Migrationen von Bundesbehörden
> – Standardarbeitsplatz
> – Beseitigung von Migrationshemmnissen"

Des weiteren hat das BMI mit IBM einen Rahmenvertrag abgeschlossen (vgl. KBSt 2002). Dieser dient unter anderem zur Förderung von OSS-Pilotprojekten und ermöglicht Behörden die günstige Beschaffung von IBM-Hardware mit OSS.

Die Bundesverwaltung verfolgt das Prinzip Interoperabilität und offene Standards nicht ausschließlich über OSS. Beispielsweise existiert auch mit Microsoft ein Rahmenvertrag (vgl. KBSt 2003*a*), in dem bestimmte Vereinbarungen festgelegt wurden.

3 Bei Sicherheitsproblemen einer proprietären Komponente kann man beispielsweise diese nicht einfach mit einer Alternative eines Konkurrenzproduktes tauschen.

Jürgen Gallmann, der Vorsitzende der Geschäftsführung von Microsoft Deutschland, meint in KBSt (2003*a*) dazu:

> „In den Vereinbarungen verpflichtet sich Microsoft dazu, die Offenlegung von Schnittstellen und Datenformaten sowie die Nutzung offener Standards in Microsoft-Produkten voran zu treiben. Dies gibt den Behörden künftig größere Flexibilität bei der Gestaltung ihrer Software–Landschaft."

Auch wenn von verschiedenen Seiten Interoperabiliät gefördert wird, bleibt zu klären, inwieweit OSS in der Lage ist, den technischen Ansprüchen einer komplexen, vernetzten IT-Infrastruktur großer Institutionen gerecht zu werden.

3. Hohe Anforderungen an IT-Netze

In vernetzten IT-Systemen muss die Server-Seite sämtliche Dienste für die Clients zur Verfügung stellen. Ein klassischer Heimcomputer kann dafür nicht verwendet werden – meist sind Großrechner bzw. Cluster-Systeme im Einsatz. Es müssen zum Beispiel Datenbank-, Web- und Applikationsserver bereitgestellt werden. Hinzu kommen Datei-, Druck-, klassische Netzwerk-[4] und Verzeichnisdienste, aber auch E-Mail- und Groupware-Lösungen sowie Proxy-Server bzw. Firewall-Systeme, Software-Verteilungs-Server, Datensicherungssysteme und viele andere mehr. Zudem müssen diese Systeme auf Grund ihres geschäftskritischen Einsatzgebietes hochverfügbar aufgebaut sein. Dabei werden so genannte Hochverfügbarkeits-Lösungen, auch *High-Availability*- (HA) Lösungen genannt, deren grundlegendes Prinzip Redundanz ist, verwendet.[5] Diese Redundanz kann durch die folgenden Modelle erreicht werden (vgl. KBSt et al. 2003, S. 287):

Failover:
: Es stehen ca. zwei bis drei Maschinen für einen Dienst bereit, wobei er bei einem Ausfall zunächst neu gestartet wird. Misslingt dies, wird der Dienst auf einen anderen Rechner transferiert.

Application Clustering:
: Es handelt sich hierbei um eine Applikation, die auf mehreren Rechnern zeitgleich arbeiten kann, sodass der Ausfall eines Rechners transparent verkraftet wird.

Server-Farmen:
: Dabei sind mehrere Server-Systeme im Einsatz, die jeweils den gleichen Dienst anbieten. Anfragen an den Dienst werden entsprechend zur Leistungsoptimierung auf die einzelnen Systeme verteilt.

4 Zu den klassischen Netzwerkdiensten zählen unter anderem Dienste, die das *Dynamic Host Configuration Protocol* (DHCP) oder das *Domain Name System* (DNS) unterstützen.

5 KBSt et al. (2003) zeigen unter anderem solche Basis-Software-Komponenten für Server-Systeme.

Es ist offensichtlich, dass neben der Anforderung an Interoperabilität die oben genannte Anforderung an Hochverfügbarkeit der unterschiedlichsten Dienste umgesetzt sein muss, um ein leistungsfähiges und produktives IT-Netz zur Verfügung zu stellen. Die Lösungen dafür sind enorm komplex.

3.1. OSS in komplexen IT-Infrastrukturen

Die Herausforderungen solch komplexer Lösungen können auch mit Freier Software bzw. Open-Source-Software bewältigt werden.

Für die im vorherigen Abschnitt genannten Herausforderungen an komplexe IT–Infrastrukturen steht auch solche Software mit ihren Vorteilen zur Verfügung, wie unter anderem die folgende kleine Auswahl an OSS-Projekten[6] zeigt:

MaxDB ist ein auf der *Structured Query Language* (SQL) basierendes relationales Datenbank-Management-System für den Einsatz mit Unternehmensanwendungen, das aus einer Allianz von MySQL AB und SAP entstanden und zudem von SAP zertifiziert ist (vgl. MySQL AB 2005).

Apache HTTP Server ist ein sehr effizienter, sicherer und erweiterbarer Webserver, der als der populärste im Internet gilt, da ihn ca. 67 % (Stand: 2004) aller Webseiten nutzen (vgl. Documentation Group 2004).

JBoss Application Server ist ein zertifizierter J2EE-Applikationsserver, der von vielen Java-Entwicklern für verteilte Systeme genutzt wird (vgl. The Professional Open Source Company 2004).

Kolab ist eine auf Freier Software basierende Groupware-Lösung, die von den unterschiedlichsten Clients genutzt werden kann (vgl. Kolab Project 2005).

Samba stellt Datei- und Druckdienste in einem Netzwerk bereit (vgl. Samba Team 2005).

ISC DHCP ist eine DHCP-Implementierung (vgl. Internet Systems Consortium, Inc. 2004*a*).

Berkeley Internet Name Daemon (BIND) ist eine Implementierung des DNS (vgl. Internet Systems Consortium, Inc. 2004*b*).

OpenLDAP ist eine LDAP-Implementierung (vgl. Zeilenga 2005).

SmoothWall ist ein Firewall-System mit benutzerfreundlicher Oberfläche (vgl. The SmoothWall Open Source Project 2005).

High-Availability Linux Project ist ein Projekt, mit dem Ziel eine hochverfügbare, zuverlässige Cluster-basierte HA-Lösung für Linux zu etablieren (vgl. Robertson 2004).

6 Weitere Open-Source-Projekte finden sich unter anderem im Internet unter http://sourceforge.net oder http://freshmeat.net.

Auf Grund solch verfügbarer Lösungen ist es möglich, alte, proprietäre IT-Systeme auf OSS umzustellen. Dies wird in der öffentlichen Verwaltung und in Unternehmen nicht nur angestrebt, sondern auch umgesetzt, wie diverse Migrationsprojekte[7] zeigen. Solche Migrationen sind allerdings nicht ohne Herausforderungen.

4. Hemmnisse bei Migrationen

Migrationen sind auf Grund von Altlasten äußerst problematisch. Meist befindet sich eine Institution in einer gewissen Abhängigkeit eines Herstellers proprietärer Produkte, weil dieser auf offene Standards verzichtet. Ein Beispiel aus dem Bereich Client-Anwendungen sind die Standarddateiformate (DOC, XLS, PPT) des MS-Office–Pakets. Auch wenn mittlerweile andere Office-Programme (z. B. OpenOffice) diese Formate öffnen können, benötigt man zur hundertprozentig korrekten Darstellung das Microsoft-Produkt, da andere Hersteller auf Grund der verborgenen Spezifikationen ihre Produkte nicht optimal anpassen können. Extrem kompliziert gestaltet sich dieser Umstand, wenn z. B. in MS-Word-Dokumenten Makros (basierend auf der wiederum proprietären Microsoft-Technologie Visual Basic, die nur auf Microsoft–Plattformen unterstützt wird) verwendet werden. In KBSt et al. (2003, S. 233) wird dazu festgestellt:

> „Im allgemeinen erfolgt die Konvertierung in einer akzeptablen Qualität, sofern es sich nicht um komplexe Dokumente mit Makros, und speziellen Format-Features handelt. Hier gibt es einige Layouteigenschaften und Formatierungsattribute in MS Office, die in OOo/SO nicht unterstützt oder anders behandelt werden. Infolgedessen ist es erforderlich, die durchgeführte Konvertierung in einem gewissen Grad manuell nachzubearbeiten, um ein dem Ausgangsdokument entsprechendes Format zu erhalten."[8]

Daraus resultierend wünschen sich Behörden

> „[. . .] die Festlegung offener Standards im Officebereich (Extensible Markup Language statt Winwordformate und Konvertierungs-, Filterprogramme für alte Officedokumente) bzw. verbindliche Vorgaben (weiter Verbreitungsgrad) für den Einsatz von OSS [. . .]",

um Migrationshemmnisse zu beseitigen. Dies haben die Umfrageergebnisse in BSI (2003, S. 9) gezeigt.

Auch auf der Server-Seite lässt sich schnell ein Beispiel für ein Migrationshemmnis finden: Active Directory. Bei dieser Microsoft-Technologie handelt es sich um eine Software, die Verzeichnisdienstfunktionalitäten zur Verfügung stellt und sich nach KBSt et al. (2003, S. 136)

7 Siehe z. B. in Schwäbisch Hall (vgl. Wilkens 2002) oder bei der Norddeutschen Affinerie AG (vgl. Ziegler 2004*b*).

8 OOo steht für die Office-Applikation OpenOffice.org und SO für das auf dem Quellcode von OOo basierende StarOffice.

> „[...] an den X.500 Standard anlehnt und via LDAP (Lightweight Directory Access Protocol) administriert werden kann."[9]

Aber dadurch, dass Active Directory nach KBSt et al. (2003, S. 148)

> „[...] eine Vielzahl von Technologien und Funktionalitäten, die es prinzipiell erleichtern, neue Funktionen und/oder effiziente Betriebsverfahren in IT Landschaften auszurollen"

bietet, kommt es auch zu Nachteilen:

> „Die Abhängigkeit von Microsoftprodukten bzw. -technologien steigt in solchen Fällen an."

Vermeidet man die oben genannten zusätzlichen Technologien bzw. Funktionalitäten, so verhindert dies nach KBSt et al. (2003, S. 150)

> „[...] in der Regel die maximale Effizienz, die mit einem Active Directory erreicht werden kann. Dies ist der Preis für eine erhöhte Unabhängigkeit."

Die Altlasten proprietärer Systeme können somit hohe Wechselkosten bedeuten. Durch den Aufbau bzw. die jahrelange Nutzung solcher proprietären Schnittstellen bzw. Produkte, lässt sich auch behaupten, dass sich Unternehmen und Behörden diese Wechselkosten angespart haben. Sind solche Wechselkosten, die beispielsweise durch den Konvertierungsaufwand von alten Datenformaten in die neuen des einzuführenden Produkts oder ganz einfach durch Schulungskosten ausgelöst werden, für einen Kunden proprietärer Produkte zu hoch, so resultiert dies in einem so genannten Lock-in (vgl. Leiteritz 2002, S. 34 ff.), da eine Migration unmöglich ist. Sollte es dennoch theoretisch gelingen, solche Aufwände zu minimieren, könnten immer noch Netzeffekte die Migration verhindern. Oftmals wird ein Anwender auf Grund von Netzeffekten dazu gezwungen, das proprietäre Produkt beizubehalten, da für das neue Datenformat keine Kommunikationspartner existieren, die damit arbeiten können.

Es ist oft zu hören, dass eine Umstellung auf Open-Source-Software teurer sei, als eine Migration auf eine neue Produktversion, wie auch in Microsoft (2005) zu lesen.

Auf Grund der oben genannten Wechselbarrieren ist eine Umstellung auf eine proprietäre Alternative ebenfalls mit hohen Kosten verbunden, die unter Umständen genauso hoch oder höher wie bei einer OSS-Migration sein können. Installation des neuen Systems, Umstellung alter Formate und Schulungen bedeuten immer Aufwände und Kosten, egal in welche Richtung die Umstellung getätigt wird. Kurzfristig gesehen kann somit die Umstellung wegen der Wechselkosten auf OSS (oder einer anderen Alternative) teurer sein. Wird ein Migrationsprojekt allerdings mit Sorgfalt geplant und durchgeführt, so kann man langfristig Einsparungen unter anderem durch wegfallende Lizenzkosten oder Nutzung alter Hardware erreichen, wie auch Schuler zur Migration der Stadt Leonberg feststellt (zitiert in Wilkens 2004):

9 X.500 und LDAP sind offene Standards.

> „Durch den Wegfall der hohen Lizenzgebühren und die Weiterverwendung der stadteigenen Computer rechnet sich diese Umstellung bereits im ersten Jahr."

Zum Abschluss dieses Abschnitts seien weitere Hinweise über Migrationshemmnisse (zumindest in der Bundes-, Landes- und Kommunalverwaltung) zusammengetragen, die aus BSI (2003, S. 8) entnommen sind:

- fehlende Akzeptanz durch den Anwender
- Schulungsaufwand für Administratoren, Nutzer, IT-Sicherheitsbeauftragte, Datenschutzbeauftragte (kein eigenes Schulungspersonal)
- Wirtschaftlichkeitsaspekte bzw. Investitionssicherung
- Einsatz vieler Windows- oder Sonderanwendungen, die zu migrieren wären
- fehlende Kompatibilität von OSS mit MS-Produkten (Word, Excel, Access)
- hoher Erprobungs- / Integrationsaufwand
- fehlende Groupwaresysteme
- fehlende Software (auch kommerzielle) für OSS-Plattformen

Hierbei handelt es sich um die meist genannten Hemmnisse. Es ist zu sehen, dass Sonderanwendungen (Fachanwendungen) zu diesen Migrationshemmnissen zählen.

5. Fachanwendungen

Unter Fachanwendungen versteht man Software, die extra zur Unterstützung eines Fachverfahrens entwickelt wurde. Fachverfahren findet man in Organisationen mit sehr speziellen Aufgaben, die sich nicht mit Standard-Software lösen lassen.

Bei Migrationsprojekten ist die Umstellung von Fachanwendungen eine der größten Herausforderungen (vgl. Vogel 2004). Fachanwendungen haben nicht so einen großen Absatzmarkt wie Standard-Software, da sie in speziellen Bereichen (Nischen) eingesetzt werden. Oftmals gibt es für Software eines Fachverfahrens nur wenige oder sogar nur einen Hersteller. Hinzu kommt, dass Fachanwendungen meist nur für eine Plattform und nicht plattformunabhängig[10] entwickelt wurden, wie auch Vogel (2004) anmerkt:

> „Die sehr knapp bemessenen Etats der Hersteller, die weder die Personalkapazitäten noch die Marktmacht haben, um im Client-Bereich neben der Microsoft-Welt noch ein zweites Betriebssystem unterstützen, hemmen den Willen eines Umstiegs."

10 Fachanwendungen sind oftmals zu sehr mit Microsoft-Technologien bzw. mit den Windows-Betriebssystemem verknüpft.

Eine Umstellung der Fachanwendungen auf eine andere Plattform, z. B. auf Linux, ist direkt nicht möglich.

Eine Initiative, die sich diesem Problem widmet, ist Linux Kommunale[11] (vgl. Kuri 2004*a*). Diese Initiative wurde von Hewlett Packard (HP) und Novell im Oktober 2004 gegründet. Unterstützt werden die Unternehmen durch die Wirtschaftsförderung der Region Stuttgart und dem Deutschen Städte- und Gemeindebund. Bei der Zusammenarbeit geht es darum, den Kommunen eine komplette, kostengünstige OSS-Lösung anzubieten, wobei die Fachanwendungen im Mittelpunkt stehen. Im Rahmen der Initiative stellen HP die Hardware und Novell mit dem SuSE Linux Enterprise Server das Betriebssystem. Ausgewählte Softwareentwickler helfen dabei, Fachanwendungen (z. B. Melde- oder Haushaltswesen) auf Linux zu portieren (siehe „Was ist Linux Kommunale?“ in www.d-mind.de (2004*b*)), wobei diese selbst nicht OSS sein muss. Eine Reihe von Fachanwendungen (z. B. Hundesteuer, Formularwesen) existieren bereits für Linux, wie in www.d-mind.de (2004*a*) gezeigt wird.

5.1. Open-Source-Fachanwendungen

Es gibt gute Gründe, Fachanwendungen nicht nur auf eine OSS-Plattform wie Linux zu migrieren, sondern diese auch als Open-Source-Software zu implementieren bzw. implementieren zu lassen. Proprietäre Fachanwendungen eines Herstellers können im Falle einer Insolvenz einen weiteren Nachteil für Unternehmen und Behörden bedeuten. Ein anderer Dienstleister kann auf Grund der verborgenen Quellen die Fachanwendung nicht weiter betreuen, d. h. Support liefern oder bei Bedarf die Anwendung weiterentwickeln.

Nutzer dieser Fachanwendungen können sich für solch einen Fall mit Hilfe von so genannten *Escrow*-Verträgen absichern, wie auch Idler et al. (2004) beschreiben. Bei Abschluss eines solchen Vertrages wird festgelegt, dass der Quelltext des Software–Produkts bei einem unabhängigen Dritten (z. B. Notar) hinterlegt wird. Weiterhin wird festgeschrieben, dass z. B. bei einer Insolvenz der Quelltext dem Kunden zugesprochen wird. Eine solche Dienstleistung ist jedoch in der Literatur der Juristen umstritten und mit Kosten verbunden. Auf Grund der zusätzlichen Kosten erscheint eine OSS-Implementierung als die optimalere Lösung.

6. Migrations-Lösungen in der Praxis

Trotz der oben genannten Herausforderungen, Probleme bzw. Hemmnisse bei Migrationen gibt es genügend Gründe, die den Einsatz von OSS motivieren. Wie auch die Artikel dieses Kapitels zeigen werden, zählen unter anderem Herstellerunabhängigkeit bzw. Vermeidung von Monokulturen, Einsparungen, Datensicherheit, Interoperabilität und sogar Förderung des Wettbewerbs dazu.

Will man eine klassische Windows-Infrastruktur auf Linux migrieren, so hat man verschiedene Möglichkeiten, das Problem mit den Fachanwendungen in den Griff zu

11 Die Website der Initiative findet sich unter: http://www.linux-kommunale.de/.

bekommen, wenn man auf diese nicht verzichten kann oder es kein Äquivalent auf der neuen Plattform gibt. Dazu gehören:

- Emulatoren
- Terminalserver
- Neuprogrammierung

Es existieren im OSS-Umfeld Emulatoren (wie z. B. „Wine" für Windows-Programme oder „DOSBox" für alte MS-DOS-Anwendungen)[12], mit denen man die benötigte Plattform für eine umzustellende Fachanwendung unter Linux emulieren kann. Für komplexe Programme, wie große Fachanwendungen, sind diese Emulatoren jedoch oftmals nicht ausgereift.

Ein anderer, indirekter, Weg kann mit Hilfe einer Terminalserver-Lösung gegangen werden. Dabei wird die Fachanwendung zentral auf einem leistungsstarken Server (z. B. mit Windows 2003 als Betriebssystem) ausgeführt, die Bildschirmausgabe wird allerdings auf den Client-Rechner (der z. B. Linux installiert hat) geschickt, an dem die Fachanwendung genutzt werden soll. Der Client-Rechner benötigt zur Darstellung der Bildschirmausgabe eine entsprechende Client-Komponente. Wie solch eine Lösung aussehen kann, zeigt Horst Bräuner in seinem Artikel „Linux im Rathaus". In dem dort beschriebenen Migrationsprojekt wird ein „universeller Client" im Zusammenhang mit einer bestimmten Terminalserver-Technik (basierend auf *Secure Socket Layer* und dem X11-Protokoll) beschrieben.

Die radikalste Möglichkeit, Fachanwendungen auch auf einem neuen System zu nutzen, ist die Neuprogrammierung dieser Software. Ein solch hoher Aufwand wird tatsächlich betrieben. Dies zeigt unter anderem der Artikel von Christiane Kunath „Migration bei der BStU auf Linux-Netware/Windows XP". Sinnvollerweise hat man sich in dem dort beschriebenen Projekt darauf festgelegt, dass die Fachanwendungen plattformneutral (auf Java-Technologien basierend) mit Browser-basierter Benutzerschnittstelle (um eine Vielzahl unterschiedlicher Client-Systeme zu bedienen) umgesetzt werden.

7. OS-Strategie, Migrationen und Entwicklungen: Die Fallbeispiele

Der erste, von Joachim Sturm verfasste Artikel in diesem Kapitel beschreibt die Open-Source-Strategie der öffentlichen Verwaltung. Auch wenn innerhalb des Artikels verschiedene Fälle für den Einsatz von OSS angesprochen werden, kann dieser Artikel nicht als direktes Fallbeispiel gelten. Dennoch macht er deutlich, welch entscheidende Rolle Open-Source-Software für die Verwaltung spielt.

Die Artikel „Migration bei der BStU auf Linux-Netware/Windows XP" von Christiane Kunath und „Linux im Rathaus" von Horst Bräuner im Anschluss daran

12 Die Projektwebseiten dieser Emulatoren lauten http://www.winehq.com/ für „Wine" bzw. http://dosbox.sourceforge.net/ für „DOSBox".

können als erfolgreiche Belege für diese Strategie angesehen werden. Es wird nicht nur gezeigt, wie in der jeweiligen Institution mit den Fachanwendungen umgegangen wird – die Artikel beschreiben auch den Einsatz von unverzichtbaren Applikationen in komplexen IT-Infrastrukturen, wie z. B. Verzeichnisdienst oder Groupware. Beim Projekt „Linux im Rathaus" wird beispielsweise der OSS-basierte Verzeichnisdienst OpenLDAP in Kombination mit dem Datei-Server Samba zum Zwecke einer zentralen Verwaltung des IT-Systems der Stadt genutzt.

Die Applikation Verzeichnisdienst ist auch ein Kernpunkt der BStU-Migration, in der auf die Infrastrukturalternative Linux-Netware/Windows[13] umgestellt wurde. Bei dieser Umstellung war auch die Einführung einer neuen Groupware-Lösung für den Endanwender entscheidend.

Das Kapitel „Fallbeispiele" beschäftigt sich aber nicht ausschließlich mit Projekten der öffentlichen Verwaltung. Auch in Unternehmen gibt es immer mehr Migrationen auf und Entwicklungen mit OSS. Dies wurde unter anderem durch den *Open Source Best Practice Award* der Öffentlichkeit näher gebracht.

Der *Open Source Best Practice Award* wurde von der Lightwerk GmbH in Kooperation mit dem Fraunhofer IAO und dem Linux-Verband durchgeführt, wie Lightwerk GmbH (2004*b*) erklärt. Es wurden die erfolgreichsten OSS-Projekte im deutschsprachigen Raum gesucht. Im Rahmen der Veranstaltung „Strategisches Open Source Symposium (SOSS)" im September 2004 hat man die Sieger vorgestellt (vgl. Lightwerk GmbH 2004*c*). In der Katagorie „Freie Wirtschaft"[14] sah die Platzierung wie folgt aus (Lightwerk GmbH 2004*b*):

1. Platz: Schering Aktiengesellschaft, Corporate Pharmacogenomics, Berlin:
„GENOMatch", Dienstleister: Tembit Software GmbH

2. Platz: Raiffeisen Zentral Bank Austria und Raiffeisen Informatik:
„Serviceorientiertes Framework für Java-basierte Portal-/ Web-Anwendungen auf Basis von Cocoon", Dienstleister: S&N AG

3. Platz: Norddeutsche Affinerie AG:
„Migration der Datei- und Verzeichnisdienste auf Linux / Samba / OpenLDAP", Dienstleister: inmedias.it GmbH

3. Platz: Mohn Media – Mohndruck GmbH:

13 Server-seitig sollen die Betriebssysteme Linux und Novell Netware (wobei sich Novell durch die Übernahme des Linux-Anbieters SuSE für Open-Source-Software geöffnet hat) eingesetzt werden (vgl. Diedrich 2003). Die Clients sollen mit Windows betrieben werden.

14 Daneben gab es noch die Kategorie „Öffentlicher Bereich" mit folgender Platzierung (Lightwerk GmbH 2004*b*): 1. Platz: Stadt Mülheim an der Ruhr: „HelpDesk-Lösung auf Basis von OTRS (Open Ticket Request System)", Dienstleister: OTRS GmbH; 2. Platz: World Health Organisation (WHO): „Distributed Digital Library – Weltweit schnell zugängliche, kostengünstige und benutzerfreundliche HIV/AIDS Toolkits für die WHO", Dienstleister: Infonoia SA; 3. Platz: Oberfinanzdirektion Hannover: „Kostengünstige Bildschirmarbeitsplätze für Sehbehinderte und Blinde auf Linux-Basis", Dienstleister: Siemens Business Services C-LAB; 3. Platz: IHK für München und Oberbayern und Landeshauptstadt München, Referat für Gesundheit und Umwelt: „Internet Map-Server (IMS)".

„Zentral regulierte aschinen-Fernwartung auf Open Source Basis“, Dienstleister: PerFact Innovation

Für die Bewertung der Projekte wurden von der Jury die folgenden Kriterien verwendet, wie in Lightwerk GmbH (2004*a*) nachzulesen ist: Innovationsgrad, Einsparungen/Verbesserungen, Relevanz/Bedeutung, Einsatz von Open Source und auch die Qualität der Dokumentation.

Der Redaktion dieses Jahrbuchs ist es gelungen, Verantwortliche bzw. Beteiligte des ersten („GENOMatch“) und des dritten („Migration der Datei- und Verzeichnisdienste auf Linux / Samba / OpenLDAP“) Platzes als Autoren zu gewinnen.

Bemerkenswert an der Lösung der Nordeutschen Affinerie AG, welche von Jörg Meyer und Carsten Brunke beschrieben wird, ist die hohe Verfügbarkeit des Systems von 99,7 % und, dass man lediglich fünf Tage für das Beseitigen von Fehlern und Störungen benötigte, obwohl 90 Tage eingeplant waren.

Beim Siegerprojekt, über das Broder Schümann und Denis Petrov berichten, handelt es sich um die Entwicklung eines verteilten IT-Systems basierend auf Open–Source-Software zur Unterstützung von Arbeitsabläufen der pharmakogenetischen Forschung. Die Einhaltung des Datenschutzes steht dabei im Vordergrund – der rechtliche Datenschutz wird bei diesem System mit Hilfe von technischem Datenschutz gesichert.

8. Fazit: Fortschreitende Etablierung von OSS

Die Artikel des Kapitels „Fallbeispiele“ zeigen deutlich, inwieweit sich, trotz aller Probleme, Open-Source-Software erfolgreich in geschäftskritischen Bereichen einsetzen lässt. Geschäftskritische Aufgaben lassen sich nur mit qualitativ hochwertiger Software unterstützen. Neben den Ergebnissen der folgenden Migrationsprojekte bescheinigt insbesondere das Ergebnis der Software-Evaluierung und die Software-Lösung des GENOMatch-Projekts die hohe Qualität. Dabei geht es hauptsächlich um verteilte Systeme, die auf offenen Standards und Internet-Technologien basieren.

Auch klassische Client-Anwendungen aus dem Open-Source-Bereich gewinnen stetig an Qualität, sodass auch über mehr Migrationen im Desktop-Bereich nachgedacht wird. Als eines der jüngeren Beispiele sei hier die Version 1.0 des Webbrowsers Mozilla Firefox der Mozilla Foundation genannt. Dieses Programm konnte innerhalb eines Monats nach seiner Veröffentlichung dem Produkt Internet Explorer des Marktführers Microsoft Marktanteile abnehmen (Tendenz steigend), wie auch Ziegler (2004*a*) berichtet.

Da Open-Source-Software ausreichend Qualitätsansprüche befriedigen kann, gibt es weitere Migrationsinteressierte. Die öffentliche Verwaltung scheint hier eine Vorreiterrolle einzunehmen. Neben Schwäbisch Hall und anderen kleineren Kommunen hatte sich die Stadt München zu einer Linux-Migration bekannt. Bereits im Mai 2003 hatte Wilkens (2003) über die Einführung von Linux berichtet. Mitte 2004 wurde das Konzept für das Projekt LiMux, bei dem es um die Umstellung der Arbeitplatz-PCs von ca. 16 000 Mitarbeitern geht, abgesegnet (Kuri 2004*b*). Das Projekt wurde aller-

dings nicht nur durch positive Schlagzeilen begleitet. Man befürchtete ein Scheitern durch mögliche Patentverletzungen (vgl. Löding 2004). Das Projekt wurde zeitweilig ausgesetzt. Demnach existiert durch die Unsicherheit auf dem Gebiet der Softwarepatente ein weiteres Migrationshemmnis.[15] Im August 2004 wurde nach Krempl und Kuri (2004*b*) das Projekt wieder aufgenommen und durch ein Rechtsgutachten im September bestätigt (vgl. Krempl und Kuri 2004*a*).

Der jüngste Bekenner zu Linux ist die Verwaltung der Stadt Wien.[16] Die Mitarbeiter der Stadt können seit diesem Jahr eine Microsoft-Plattform oder die spezielle Linux-Distribution Wienux (enthält unter anderem Firefox und OpenOffice) wählen (Rindl 2005).

Auf Grund des Vorzeigecharakters dieser beiden Projekte könnten bereits zwei neue Fallbeispiele für das Open Source Jahrbuch 2006 gefunden sein.

Literatur

BSI (2003), 'Erhebung zur Nutzung von Freier Software in der Bundes-, Landes- und Kommunalverwaltung – Enquete 2003', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/Anlage304735/K:KBStWebsiteRedaktionenquet%202003.pdf [30. Jan 2005].

Diedrich, O. (2003), 'Netzwerkspezialist Novell kauft Linux-Distributor SuSE [2. Update]', heise online, http://www.heise.de/newsticker/meldung/41684 [30. Jan 2005].

Documentation Group (2004), 'Welcome! – The Apache HTTP Server Project', Apache Software Foundation, http://httpd.apache.org/ [30. Jan 2005].

Idler et al. (2004), 'Escrow', Wikipedia – Die freie Enzyklopädie, http://de.wikipedia.org/wiki/Escrow [30. Jan 2005].

Internet Systems Consortium, Inc. (2004*a*), 'Dynamic Host Configuration Protocol (DHCP)', Internet Systems Consortium, Inc., http://www.isc.org/index.pl?/sw/dhcp/ [30. Jan 2005].

Internet Systems Consortium, Inc. (2004*b*), 'ISC BIND', Internet Systems Consortium, Inc., http://www.isc.org/index.pl?/sw/bind/ [30. Jan 2005].

KBSt (2002), 'Schily zieht Jahresbilanz: Open Source Software in der öffentlichen Verwaltung – Auf dem Weg zu einer vielfältigen und offenen Software-Landschaft', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/Software/-,72/IBM-Kooperation.htm [30. Jan 2005].

KBSt (2003*a*), 'Neue Lizenzrahmenverträge mit Microsoft', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/Software/-,73/Microsoft.htm [30. Jan 2005].

15 Weitere Information zur Problematik der Softwarepatente gibt es im Artikel von Stefan Krempl „Der Kampf gegen Softwarepatente – Open Source im Auge des Sturms" im Kapitel „Recht und Politik".

16 Stand: 23. Januar 2005

KBSt (2003*b*), 'Software für die Bundesverwaltung', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/-,56/Software.htm [30. Jan 2005].

KBSt (2004), 'Open Source Software für die Bundesverwaltung', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/Software/-,74/Open-Source.htm [30. Jan 2005].

KBSt, BSI, Bundesverwaltungsamt und C_sar AG (2003), 'Migrationsleitfaden – Leitfaden für die Migration der Basissoftwarekomponenten auf Server- und Arbeitsplatz-Systemen', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/Anlage304426/Migrationsleitfaden.pdf [30. Jan 2005]. Version 1.0; Schriftenreihe der KBSt, ISSN 0179-7263, Band 57.

Kharitoniouk, S. und Stewin, P. (2004), Kapitel 1 Grundlagen und Erfahrungen – Einleitung, *in* R. A.Gehring und B. Lutterbeck (Hrsg.), 'Open Source Jahrbuch 2004: Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns, Berlin, S. 1–15.

Kolab Project (2005), 'The Kolab Project :: Home', The Kolab Project, http://kolab.org/ [30. Jan 2005].

Krempl, S. und Kuri, J. (2004*a*), 'Rechtsgutachten bestätigt München in seinem Linux-Kurs', heise online, http://www.heise.de/newsticker/meldung/51022/ [30. Jan 2005].

Krempl, S. und Kuri, J. (2004*b*), 'Stadt München setzt Linux-Migration fort', heise online, http://www.heise.de/newsticker/meldung/49979 [30. Jan 2005].

Kuri, J. (2004*a*), 'Hewlett-Packard und Novell starten „Linux Kommunale"', heise online, http://www.heise.de/newsticker/meldung/52589 [30. Jan 2005].

Kuri, J. (2004*b*), 'Münchner Stadtrat segnet Konzept zur Linux-Migration ab', heise online, http://www.heise.de/newsticker/meldung/48313 [30. Jan 2005].

Leiteritz, R. (2002), 'Internet Ökonomie', Technische Univerität Berlin, Institut für Wirtschaftsinformatik, Informatik und Gesellschaft, http://ig.cs.tu-berlin.de/oldstatic/w2002/ir1/007/IR1-Ecomm-Release3-Folien.pdf [30. Jan 2005]. Vorlesung: Information Rules 1 WS 2002/2003.

Lightwerk GmbH (2004*a*), 'Bewertungskriterien', Lightwerk GmbH, http://soss.lightwerk.com/content/award/kriterien/index_ger.html [30. Jan 2005].

Lightwerk GmbH (2004*b*), 'Open Source Best Practice Award', Lightwerk GmbH, http://soss.lightwerk.com/content/award/index_ger.html [30. Jan 2005].

Lightwerk GmbH (2004*c*), 'SOSS', Lightwerk GmbH, http://soss.lightwerk.com/content/index_ger.html [30. Jan 2005].

Löding, T. (2004), 'München legt Linux-Projekt wegen der Softwarepatente auf Eis', heise online, http://www.heise.de/newsticker/meldung/49735 [30. Jan 2005].

Microsoft (2005), 'Fakten zu Windows und Linux – Analysen und Studien', Microsoft Corporation, http://www.microsoft.com/germany/diefakten/studien.mspx [30. Jan 2005].

MySQL AB (2005), 'MySQL Products', MySQL AB, http://www.mysql.com/products/ [30. Jan 2005].

Rindl, M. (2005), 'Wienux, das Linux für Wien', heise online, http://www.heise.de/newsticker/meldung/55434 [30. Jan 2005].

Robertson, A. (2004), 'Linux-HA Project Web Site', High-Availability Linux Project, http://www.linux-ha.org/ [30. Jan 2005].

Samba Team (2005), 'Opening Windows to a Wider World', samba.org, http://us4.samba.org/samba/ [30. Jan 2005].

The Professional Open Source Company (2004), 'JBoss Application Server', JBoss.com, http://www.jboss.org/products/jbossas [30. Jan 2005].

The SmoothWall Open Source Project (2005), 'Welcome! - SmoothWall.org', The SmoothWall Open Source Project, http://www.smoothwall.org/ [30. Jan 2005].

Vogel, M. (2004), 'Fachanwendungen Hauptproblem bei Linux-Migration', Pro-Linux News, http://www.pro-linux.de/news/2004/6451.html [30. Jan 2005].

Voß et al. (2004), 'Interoperabilität', Wikipedia – Die freie Enzyklopädie, http://de.wikipedia.org/wiki/Interoperabilit%C3%A4t [30. Jan 2005].

Wheeler, D. A. (2005), 'Why Open Source Software / Free Software (OSS/FS, FLOSS, or FOSS)? Look at the Numbers!', David A. Wheeler's Personal Home Page, http://www.dwheeler.com/oss_fs_why.html [30. Jan 2005]. Revised as of January 29, 2005.

Wilkens, A. (2002), 'Linux kommt nach Schwäbisch Hall', heise online, http://www.heise.de/newsticker/meldung/32640 [30. Jan 2005].

Wilkens, A. (2003), 'Münchener Rathaus-SPD entscheidet sich für Linux [Update]', heise online, http://www.heise.de/newsticker/meldung/37126 [30. Jan 2005].

Wilkens, A. (2004), 'Leonberg geht den Linux-Weg', heise online, http://www.heise.de/newsticker/meldung/44802 [30. Jan 2005].

Zeilenga, K. (2005), 'OpenLDAP, Title', OpenLDAP Foundation, http://www.openldap.org/ [30. Jan 2005].

Ziegler, P.-M. (2004*a*), 'Firefox: 10 Millionen Downloads in einem Monat', heise online, http://www.heise.de/newsticker/meldung/54187 [30. Jan 2005].

Ziegler, P.-M. (2004*b*), 'Sieger des ersten „Open Source Best Practice Award" präsentiert', heise online, http://www.heise.de/newsticker/meldung/51885 [15. Jan 2005].

www.d-mind.de (2004*a*), 'Linux kommunale – Fachanwendungen', Wirtschaftsförderung Region Stuttgart GmbH, http://www.linux-kommunale.de/fachanwendungen.php [30. Jan 2005].

www.d-mind.de (2004*b*), 'Linux kommunale – Linux Kommunale aktuell', Wirtschaftsförderung Region Stuttgart GmbH, http://www.linux-kommunale.de/ [30. Jan 2005].

Die Open-Source-Strategie der öffentlichen Verwaltung

JOACHIM STURM

Als zentraler Punkt der Open-Source-Strategie der öffentlichen Verwaltung wird die Software-Vielfalt gesehen. Neben offenen Standards, wie z. B. der *Extensible Markup Language* (XML), werden das Dokument „Standards und Architekturen für E-Government-Anwendungen" (SAGA) und die Förderung von Open-Source-Software zum Erreichen des Ziels genutzt. Neben der Tatsache, dass Open-Source-Software einen Schritt in Richtung Software–Vielfalt darstellt, spielt in diesem Zusammenhang auch die IT-Sicherheit, die damit erreicht werden kann, eine bedeutende Rolle. Ein Mittel zur Förderung von Open-Source-Software stellt das Open-Source-Software-Kompetenzzentrum dar – dort werden relevante Informationen und Erfahrungen zum Thema für die öffentliche Verwaltung präsentiert. Die Open-Source–Strategie findet sich auch im Migrationsleitfaden wieder. Er ist für Entscheider, aber auch für IT-Experten gedacht.

1. Einleitung

Mit einer pfiffigen Idee allein lässt sich keine Software optimieren. Voraussetzung für einen nachhaltigen Innovationsprozess, an dem sich viele beteiligen, ist ein offener Zugang zum Wissen. Für die Softwareentwicklung gewährleistet der Gebrauch des Open-Source-Prinzips diese Wissenstransparenz. Interessierte können das Wissen nutzen, ergänzen und durch eigene Innovationen wirtschaftlich davon profitieren.

Die Bundesregierung unterstützt und fördert den Einsatz von Open-Source-Software (OSS) und die Verwendung offener Standards in der öffentlichen Verwaltung. In den deutschen Behörden spielt Freie Software auf Servern und Desktop-PCs eine an Bedeutung zunehmende Rolle.

Der nachfolgende Abriss über die OSS-Strategie des Bundes und dessen Projekte stellt die Maßnahmen vor, mit denen der Bund die Software-Vielfalt in den Behörden erreicht. Beleuchtet werden die Bedeutsamkeit offener Standards und Aussagen, die das Dokument „Standards und Architekturen für E-Government-Anwendungen" (SAGA) in diesem Kontext trifft, die Möglichkeiten der *Extensible Markup Language*(XML), Open-Source-Software und deren Bedeutung für die Sicherheit. Ein

Überblick zu aktuellen OSS-Projekten der Bundesverwaltung, zum virtuellen OSS–Kompetenzzentrum und dem Migrationsleitfaden schließt sich an. Am Schluss finden Sie einen Ausblick auf künftige Entwicklungen.

2. Software-Vielfalt und offene Standards

Bunt soll sie sein – die Software-Landschaft in den deutschen Behörden. Mit dieser Strategie der Software-Vielfalt verfolgt der Bund gleich mehrere Ziele: Zum Ersten gewährleistet eine bunte Software-Landschaft mehr Sicherheit, da Sicherheitsmechanismen großflächig verbreitet werden können. Zum Zweiten wirkt die Software–Vielfalt einer Monopolisierung des Marktes entgegen, da neue Anbieter eher einen Zugang zum Markt und in etablierte Netzwerke finden. Und eine bunte Software–Landschaft in den Behörden garantiert drittens Herstellerunabhängigkeit und einen fairen Wettbewerb.

Damit solch ein buntes Umfeld aus proprietärer und Open-Source-Software entstehen kann, müssen offene und frei verfügbare Standards vorhanden sein. Sie stellen als Minimalkonsens sicher, dass komplexe Systeme miteinander kommunizieren können.

3. Offene Standards

Ein wichtiges Kriterium bei der Auswahl von IT-Systemen ist demnach deren Interoperabilität. Offene Standards erfüllen dieses Kriterium – sie sind frei zugänglich, dokumentiert und einsetzbar. Sie ermöglichen es, dass eine große Produktpalette zur Verfügung steht – sowohl Open-Source-Software als auch kompatible proprietäre Software. Durch die Marktbreite von Entwicklungsfirmen, welche die offenen Schnittstellen nutzen können, erhöht sich die Verfügbarkeit der Produkte und deren Qualität.

Im Bereich der Basistechnologien haben sich offene Standards weitestgehend etabliert. Auf der Ebene der Software-Anwendungen ist diese Form der nahtlosen Zusammenarbeit noch nicht erreicht. Mit Blick auf E-Government-Anwendungen, innerhalb derer die verschiedensten Applikationen und Anwendungen miteinander und untereinander kommunizieren müssen, wurde mit dem Dokument SAGA speziell auf dafür geeignete Standards eingegangen.

4. SAGA und offene Standards

Das Dokument „Standards und Architekturen für E-Government-Anwendungen" stellt in verdichteter Form verbreitete Standards, Verfahren, Methoden und auch Produkte der modernen IT-Entwicklung für E-Government vor. Mit SAGA treibt der Bund die Standardisierung von IT-Komponenten im E-Government voran. Soweit es möglich ist, setzt SAGA auf offene Standards. Wo es noch keine verbreiteten offenen Standards gibt, werden proprietäre Standards verwendet, um eine Kommunikation sicherzustellen. Ziel ist es, durch die Verwendung offener Standards, wie z. B. *Extensible*

Markup Language oder *Online Services Computer Interface* (OSCI), zukunftsorientiert und plattformunabhängig zu agieren. SAGA in der Version 1.1 wurde im Februar 2003 erstmals publiziert. Entsprechend der Entwicklungen im IT-Bereich wird das Dokument kontinuierlich fortgeschrieben. Die aktuellste Version SAGA 2.0 erschien im Dezember 2003. Im Jahr 2005 wird es eine neue SAGA-Version geben.

SAGA hat sich nicht nur in der Bundesverwaltung als anerkannter Standard durchgesetzt. Vorteil der SAGA-Standards ist, dass die Privatwirtschaft als Partner der öffentlichen Verwaltung bereits bei den ersten Entwürfen ihre Überlegungen einbringen kann. Darüber hinaus ist SAGA in das *Interoperability Framework* im IDA-Programm der EU (IDA: *Interchange of Data between Administrations*) eingegangen.

Beispiel XML: Um die Kompatibilität und den notwendigen Datenfluss zwischen verschiedenen Anwendungen problemlos zu gewährleisten, hat sich der Einsatz des Standards XML bewährt. Dem Beispiel der Wirtschaft folgend, hat die Bundesverwaltung daher den Einsatz von Standards aus der XML-Familie zur Vernetzung ihrer E-Government-Anwendungen vorgesehen. XML soll in Zukunft auch die Interoperabilität der Office-Anwendungen verbessern.

Die *Extensible Markup Language* ist eine Technologie, die standardisierte Techniken bietet, um strukturierte, lesbare Dateien zu erstellen. Mit Hilfe von XML können somit Austauschsprachen und -formate erschaffen werden, um eine standardisierte Kommunikation zwischen verschiedenen Anwendungen zu ermöglichen. Die *Extensible Markup Language* nimmt die Rolle eines Vermittlers zwischen den verschiedenen Software-Produkten ein. Es ist mit XML beispielsweise möglich, einen standardisierten Austausch für Office-Dokumente, wie Tabellenkalkulations- oder Text-Dokumente, zu erzeugen, die von beliebig vielen Anwendungen verarbeitet werden können.

Die Bundesregierung bietet auf den Webseiten der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) den XML-Infopoint an. Dieser stellt in seiner ersten Ausbaustufe Informationen und Links rund um das Thema „Einsatz von XML" bereit und gibt einen Überblick über die aktuellen XML-Projekte in der Bundesverwaltung. Neben generellen Informationen zu XML sind dort aktuelle Nachrichten sowie eine Übersicht von bereits existierenden XML-Produkten und XML-Schemata in der Bundesverwaltung enthalten. Das Angebot des XML-Infopoints wird aktualisiert und erweitert. In einer „Erfahrungsdatenbank" können positive wie negative Erfahrungen bei der Erstellung von Austauschsprachen zugänglich gemacht werden.

5. Ein Schritt zur Software-Vielfalt: Open-Source-Software

Open-Source-Software bietet als entscheidenden Vorteil das Recht auf Verwendung und Anpassung. Bei der Beschaffung von Software sind meist Anpassungen oder ganze Neuprogrammierungen notwendig. Hier kann Open-Source-Software als Ausgangsbasis verwendet werden. So kann auf bestehende Systeme aufgebaut werden.

Open-Source-Software erfüllt zudem die Anforderung der Interoperabilität, da offene Standards verwendet werden. Die Zahl der Behörden, die ihre Systeme und Anwendungen ganz oder teilweise auf Open-Source- bzw. Freie Software umstellen,

wächst kontinuierlich. Innovative OSS-Projekte – wie zum Beispiel das Botschaftsnetz des Auswärtigen Amtes oder die elektronische Signatur für Open-Source-Systeme aus dem Bundesamt für Sicherheit in der Informationstechnik (BSI) – haben sogar außerhalb der Verwaltung die Aufmerksamkeit der Fachleute auf sich gezogen.

Zum Dritten ist der Einsatz von OSS vor allem langfristig wirtschaftlich sinnvoll. Durch die Unabhängigkeit von Herstellern wird der Einsatz der Systeme nachhaltig: Die Insolvenz eines Software-Herstellers bedeutet für den Nutzer von dessen Software heute in der Regel den Kauf und die Einführung eines neuen Software-Produkts. Wie die Praxis zeigt, müssen nicht nur kleine und mittelständische Unternehmen Insolvenz anmelden, so dass Investitionen in deren Software verloren sind. Die Lizenzen von Open-Source-Software sichern den Zugang zum Quellcode sowie das Recht, diesen zu verändern und einzusetzen. Mit dem Besitz des Quellcodes und dem Recht, diesen zu ändern und in der veränderten Form auch einzusetzen, ist ein Investitionsschutz erreicht. Gleiches gilt auch für die Sicherstellung des Support.

6. OSS und Sicherheit

Der modulare Aufbau von Open-Source-Software birgt einen wichtigen Sicherheitsaspekt: Je minimaler und übersichtlicher ein System aufgebaut ist, desto sicherer kann es gemacht und umso besser kontrolliert werden. Modularität ermöglicht es, die Funktionen zu minimieren und die Fläche für potenzielle Angriffe gering zu halten. Gleichzeitig wird die Ausfallsicherheit erhöht.

Ein Beispiel ist das Produkt SINA des Bundesamtes für Sicherheit in der Informationstechnik. SINA steht für *Sichere Inter-Netzwerk-Architektur* und gewährleistet als solche unter anderem die Kommunikation zwischen den deutschen Botschaften. Bei der Entwicklung von SINA waren die Freiheiten von Open-Source-Software richtungsweisend: SINA baut auf dem Linux-Kernel auf, der durch Minimierung und Anpassungen zu einem hochsicheren System ausgebaut wurde. Ohne die Möglichkeit, die bestehende Software als Basis zu nutzen, wäre die Entwicklung einer solchen Anwendung aus wirtschaftlichen Gründen nicht realisierbar gewesen. So waren lediglich individuelle Anpassungen notwendig.

Der Einsatz von Open-Source-Software hat durch die Freiheiten – Einsatz, Lernen, Erweitern, Verteilen – sicherheitstechnisch weitere Vorteile. Warnmeldungen über gefundene Fehler bei Sicherheitsprüfungen können veröffentlicht werden, weil es kein *Non Disclosure Agreement* gibt. Der Anwender kann so bei Sicherheitslücken schnell informiert werden und Gegenmaßnahmen ergreifen. Mit Open-Source-Software sind die Anwender in der Lage, Fehler selbst zu beheben – ein Vorgehen, dass bei proprietärer Software aus lizenzrechtlichen Gründen meist untersagt ist.

Die Prüfung von Software auf Sicherheitslücken sollte immer möglich sein, auch wenn der Hersteller dem nicht zustimmt. Beim Einsatz von Software kann dies ein K.O.-Kriterium sein. Es steht Vertrauen versus Wissen. Der Zugang zum Quellcode ist eine notwendige, wenn auch keine hinreichende Voraussetzung für die Sicherheitsüberprüfung von Software. Der dafür vorgesehene internationale Standard (*Common*

Criteria) zur Prüfung (Zertifizierung) von Software-Produkten fordert etwa für eine Reihe der vorgesehenen Prüfschritte die Offenlegung der Quellen.

Selbst wenn der öffentlichen Verwaltung der Einblick in proprietäre Software von den Herstellern manchmal gewährt wird, so ist dies ein Privileg, dass der Wirtschaft, vor allem dem Mittelstand, oft nicht zuteil wird.

Der erfolgreiche Erhalt von IT-Sicherheit bedingt die genaue Kenntnis des IT-Systems sowie eine regelmäßige Wartung und eine schnelle Behebung von Sicherheitslücken. Der Einsatz Freier Software hat in diesem Prozess strategische Vorteile, bietet jedoch nicht per se eine Gewähr für ein sicheres System.

7. Open-Source-Software-Kompetenzzentrum

Ganz im Sinne des OSS-Gedankens, Wissen frei zugänglich zu machen, wurde das virtuelle Open-Source-Software-Kompetenzzentrum (OSS-CC) im März 2004 gegründet. Das OSS-CC entstand aus einer gemeinsamen Initiative des BMI, BMVBW und BSI. Das OSS-CC ist die zentrale Informationsstelle im Netz für Fragen zum Thema Open Source innerhalb der Bundesverwaltung. In seiner derzeitigen ersten Ausbaustufe stellt das Kompetenzzentrum OSS-Projekte aus der Bundesverwaltung in einem Projektkatalog vor, benennt Ansprechpartner und liefert Informationen zu Veranstaltungen. Die Behörden erhalten die Möglichkeit, sich über Erfahrungen und Problemlösungen auszutauschen.

Derzeit (Stand: November 2004) laufen die Arbeiten an der zweiten Ausbaustufe des OSS-CC. Neben der Erweiterung und Aktualisierung des Projektkatalogs, der Studien bzw. Publikationen und der Linksammlung soll das Webangebot durch Informationen zum Thema Open Source abgerundet werden. Für Einsteiger in das Thema werden die wichtigsten Begrifflichkeiten erläutert. Dazu zählt insbesondere die Abgrenzung zu anderen Software- und die Einordnung von Lizenzmodellen. Zudem wird es eine einführende Vorstellung konkreter OSS-Komponenten als mögliche Alternative zu proprietärer Software geben. Dabei wird stets Bezug auf das Dokument „Standards und Architekturen für E-Government-Anwendungen“ genommen. In weiteren Rubriken werden Informationen und Hilfestellungen zur korrekten Formulierung von Ausschreibungen und zu rechtlichen Aspekten gegeben.

Eine wesentliche Aufgabe des OSS-CC ist die Vernetzung von Kompetenzen. Dazu wird eine Zusammenarbeit mit regionalen Kompetenzzentren und Schlüsselpersonen aus dem OSS-Umfeld angestrebt. Neben dem Aufbau eines nationalen Informations- und Kompetenznetzes sollen der Austausch und die Zusammenarbeit mit OSS-Kompetenzzentren auf europäischer Ebene forciert werden. Dadurch soll der Nutzen des Projektkataloges erweitert werden. Beispielsweise können regionale Zentren durch die Vorstellung gesammelter Erfahrungen aus OSS-Projekten auf Landes- bzw. Kommunalebene das bereits vorhandene Know-how ergänzen.

Im Endausbau soll das Open-Source-Software-Kompetenzzentrum der zentrale Informationspunkt für Behörden werden, die Open-Source-Projekte beginnen wollen. Im Zuge der Einer-für-Alle-Projekte wird hier ein Dreh- und Angelpunkt für IT-Anwendungen geschaffen.

8. Migrationsleitfaden

Geht es um die Migration von Systemen – sei es auf OSS oder proprietäre Software – bietet der vom Bundesministerium des Innern herausgegebene Migrationsleitfaden Entscheidungshilfen. Einige Behörden migrieren zur Zeit noch nicht auf Open–Source-Software, weil ihnen zu wenige migrierte Bereiche oder Behörden bekannt sind, die als Beispiel dienen und von deren Erfahrungen sie profitieren können. Eine Hilfe bieten die Pilotprojekte in Behörden, deren Ergebnisse im Migrationsleitfaden zu finden sind.

Der Leitfaden für die Migration von Basissoftwarekomponenten auf Server- und Arbeitsplatz-Systeme wurde von der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung herausgegeben. Er zeigt fallbezogene Migrationspfade mit detaillierten Informationen. Der Leitfaden richtet sich primär an zwei Zielgruppen. Einerseits an Entscheider, die für die Planung und Umsetzung der IT-Strategien und -vorhaben verantwortlich sind. Andererseits an IT-Experten, die die Entscheidungen umsetzen. In einer Bestandsaufnahme werden die üblicherweise in den Behörden eingesetzten Systeme und Programme ausführlich beschrieben und Alternativen mit ihren Vor- und Nachteilen vorgestellt. Anhand dieser Betrachtungen formuliert der Migrationsleitfaden Migrationsempfehlungen. Mittlerweile wurde der Leitfaden über 100 000 Mal von den Webseiten der KBSt heruntergeladen.

Auch der Migrationsleitfaden wird aktuell fortgeschrieben. Im Fokus der Überarbeitung wird eine stärkere Betrachtung von Windows 2003 und XP stehen. So wird die fortgeschriebene Fassung des Migrationsleitfadens eine Studie zur Interoperabilität zwischen MS Office 2003 und OpenOffice 1.1 auf Basis von XML-Standards beinhalten. Darüber hinaus werden in der Fortschreibung die Wirtschaftlichkeitsbetrachtungen überarbeitet und die rechtlichen Aspekte von Migrationen erläutert.

9. Fazit

Im internationalen Vergleich gilt Deutschland im Open-Source-Bereich als führend; die deutschen Verwaltungen haben beim Einsatz eine Vorreiterrolle übernommen. Gleichwohl befürwortet die Bundesregierung im Rahmen einer ausgewogenen Strategie Initiativen zur Verwendung offener Standards in proprietären Produkten.

Auch zukünftig wird der Bund die Strategie der Software-Vielfalt fortführen und damit Open-Source-Software unterstützen. Ziel ist es, die Handlungsfähigkeit der Verwaltung in IT-Fragen zu erhalten und die Wahlfreiheit beim Einsatz von Informationstechnik sicherzustellen und zu verbessern.

Weiterführende Informationen

Die KBSt bietet detaillierte Informationen unter den folgenden Webadressen an:

Webauftritt der KBSt: http://www.kbst.bund.de/

Open-Source-Software-Kompetenzzentrum:
http://www.kbst.bund.de/-,247/OSS-Kompetenzzentrum.htm

Migrationsleitfaden:
http://www.kbst.bund.de/Anlage304426/Migrationsleitfaden.pdf

SAGA-Dokument:
http://www.kbst.bund.de/Anlage304423/SAGA_Version_2.0.pdf

XML-Infopoint:
http://www.kbst.bund.de/E-Government/-,231/XML-Technologie.htm

Weitere Informationen zum IDA-Programm der EU sind unter der Adresse
http://europa.eu.int/ida/ abrufbar.

Migration bei der BStU auf Linux-Netware/Windows XP

CHRISTIANE KUNATH

Der vorliegende Artikel beschreibt das Migrationsprojekt der Behörde BStU. Dieses Projekt unterteilt sich in zwei Unterprojekte. Es wurden 80 Server und 2 300 Client-Arbeitsplätze migriert. Bei der Client-Migration wurde zunächst auf das proprietäre Windows XP umgestellt, da der laufende Betrieb, d. h. der Einsatz der 130 Fachanwendungen, gesichert bleiben musste. Neben dieser Richtlinie (Sicherung des laufenden Betriebs) beschreibt dieser Artikel weitere Rahmenbedingungen, die für IT-Projekte der öffentlichen Verwaltung gelten. Dazu zählen unter anderem die Software-Strategie der Bundesverwaltung (Abbau von Monokulturen), der Migrationsleitfaden der KBSt, das SAGA-Dokument und eine Wirtschaftlichkeitsbetrachtung. Der Bericht zeigt zudem, wie man anhand einer umfassenden Migrationsstudie (Ist-Aufnahme, Infrastrukturalternativen, Nutzwertanalyse, IT-Wirtschaftlichkeitsbetrachtung) zur IT-Infrastruktur Linux-Netware/Windows fand. Die Umstellung der Clients auf Open-Source-Software wird ebenfalls angesprochen. Als essentielle Voraussetzung dafür gilt die plattformunabhängige Neuprogrammierung der Fachanwendungen.

1. Einleitung

Im folgenden einleitenden Abschnitt des Migrationsberichts wird kurz eingeordnet, wer die BStU ist, wie die IT-Landschaft dieser Behörde über die Jahre gewachsen ist und welche Notwendigkeiten die Migration veranlassten.

1.1. Die Behörde der Bundesbeauftragten für die Stasiunterlagen der ehemaligen Deutschen Demokratischen Republik (BStU)

Nach Maßgabe des Gesetzes über die Unterlagen des Staatssicherheitsdienstes der ehemaligen DDR (StUG) verwahrt, verwaltet und erschließt die Bundesbeauftragte die Unterlagen des Ministeriums für Staatssicherheit (MfS) nach archivischen Grundsätzen. Sie erteilt Auskünfte, gewährt Akteneinsicht und gibt Kopien heraus. So können Opfer des Staatssicherheitsdienstes der DDR ihre Akten einsehen, um zu erfahren,

inwieweit der Staatssicherheitsdienst Einfluss auf ihr Leben genommen hat. Parlamente, Behörden, Kirchen, Unternehmen, Parteien und Verbände können Ersuchen an die Bundesbeauftragte richten, um einen im Gesetz festgelegten Personenkreis auf eine frühere Tätigkeit für das MfS überprüfen zu lassen.

Daneben unterstützt die Bundesbeauftragte Medien, Forschung und politische Bildung bei der Aufarbeitung der Tätigkeit des Staatssicherheitsdienstes, indem sie die Unterlagen Journalisten und Forschern auf der Grundlage rechtlicher Bestimmungen zur Verfügung stellt.

Zur Erforschung von Struktur, Aufgaben und Wirkungsweise des MfS unterhält die Behörde eine eigene Forschungsabteilung, deren Arbeitsergebnisse in Publikationen, öffentlichen Veranstaltungen und Dokumentationszentren präsentiert werden.

Am 30. Juni 2004 hatte die BStU ca. 2 300 Mitarbeiter, welche auf 14 Standorte verteilt sind. Somit ist die BStU eine „große Behörde".

1.2. Historie der IT-Landschaft

Bei der BStU wurden seit Ende 1991 schrittweise mehrere lokale Netzwerke (*Local Area Network* – LAN) aufgebaut. Heute[1] sind ca. 2 300 PC-Arbeitsplätze und ca. 80 Server an diesem Netz beteiligt.

Nach der Arbeitsaufnahme der Behörde im Herbst 1991 verteilten sich die Berliner Liegenschaften auf bis zu zehn Standorte, die seitdem noch häufig gewechselt wurden. Diese waren untereinander nicht vernetzt, aber alle mit Informationstechnik ausgestattet. Erst mit dem Umzug in die Otto-Braun-Straße (das Archivgebäude für die Stasiunterlagen befindet sich weiterhin in der Magdalenenstraße in Berlin-Lichtenberg) wurde der heutige Zustand der hundertprozentigen Ausstattung (d. h. alle Büroarbeitsplätze sind mit PCs ausgestattet) erreicht.

Die Außenstellen befinden sich in den ehemaligen Bezirksstädten der DDR. Die Stadt Cottbus bildet eine Ausnahme. Dort gibt es nur eine so genannte Lesestelle. Das eigentliche Archiv befindet sich in Frankfurt/Oder. Mit dem Auftrags- und Mitarbeiterrückgang erfolgt in den nächsten Jahren eine weitere Umstrukturierung und damit auch eine Reduzierung der Außenstellen. Geplant sind fünf Außenstellen mit den Archiven und fünf Lesestellen.

Die Informationstechnik der Behörde war von Anfang an durch die Server-seitig eingesetzten Betriebssysteme von Novell und Hewlett Packard (HP) sowie Client–seitig von Microsoft geprägt. Ab 1991 wurde dabei das System Novell Netware 3.11 bzw. ab 1996 die Version 4.11 als Datei-Server genutzt. Als Datenbankserver kam 1991 ein SCO-UNIX-System und seit 1992 ein HP-UX-System mit Informix DBMS zum Einsatz. Auf der Client-Seite begann man 1991 mit MS Windows 3.11 (auf MS DOS basierende grafische Benutzeroberfläche). Ab dem Jahr 1998 wurden dann das Betriebssystem MS Windows NT und die Büroapplikation MS Office 97 angewendet. Hinzu kam, dass im Laufe der Zeit für den Client ca. 130 Visual-Basic- bzw. 4GL-Applikationen entwickelt und dazu noch ca. 80 kommerzielle Software-Produkte eingesetzt wurden.

1 Stand: September 2004

Zu dieser IT-Landschaft kam durch die Einführung eines Personalmanagementsystems (EPOS) im Geschäftsbereich des Bundesministeriums des Innern (BMI) der Einsatz des Betriebssystems MS Windows NT Server 3.51 bzw. 4.0 und des Datenbankservers MS SQL Server 6.0 bzw. 6.5 hinzu.

Die für die Infrastruktur nötigen Netzwerk- und Systemmanagementsysteme arbeiten auf der Basis von HP OpenView und Cisco Works auf dem Betriebssystem MS Windows 2000 Server.

1.3. Die Notwendigkeit der Migration bei der BStU

Seit dem Jahr 2002 verfolgt die Bundesverwaltung eine Software-Strategie der offenen Standards, der Interoperabilität und der Vermeidung von Abhängigkeiten durch Monokulturen und ermöglicht somit eine stärkere Vielfalt in der Software-Landschaft der Behörden. Die bis dahin vorherrschende Microsoft-IT-Infrastruktur in vielen Behörden sollte durch Alternativen abgelöst bzw. ergänzt werden.

Unterstützt werden diese Ansätze durch den „Migrationsleitfaden" der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern (KBSt). Er bietet den Behörden strategisch-wirtschaftliche und technische Entscheidungshilfen bei einer Migration der eingesetzten IT-Systeme zu Open-Source-Software (OSS). Grundlage dafür sind einheitliche Standards und Software-Architekturen für E-Government-Anwendungen. Diese sind im für die Bundesverwaltung verbindlichen Dokument „Standards und Architekturen für E-Government-Anwendungen" (SAGA) festgelegt. SAGA erfasst Entwicklungen und Anwendungen, um Dienstleistungen möglichst einheitlich und interoperabel elektronisch abbilden zu können.

Bei der BStU gab es im Jahr 2002 zwei wesentliche Gründe für die Migration: Die Abkündigung des Supports der primär eingesetzten Betriebssysteme der Hersteller Microsoft und Novell bzw. fehlende Treiber für periphere Geräte sowie die Umkehr von der bis dahin existierenden Abhängigkeit von Microsoft-Produkten.

Zu beachtende Besonderheiten der Migration

In den Anfangszeiten der Behörde existierten für die spezifischen Anforderungen der BStU wenig geeignete kommerzielle Programme auf dem Markt. Daher wurde ein großer Teil der Programme in enger Zusammenarbeit mit dem Fachanwender erstellt. Diese in Visual Basic und 4GL selbst entwickelten Fachanwendungen haben daher eine außerordentlich hohe Bedeutung für die Arbeitsfähigkeit der Fachbereiche. Viele Aufgaben der Fachbereiche der BStU sind nur unter Einsatz dieser Anwendungen zu erfüllen. Die Verfügbarkeit der Fachanwendungen durfte daher bei und nach der Migration nicht eingeschränkt sein.

2. Ablauf der Migration

Der folgende Abschnitt beschreibt den Migrationsverlauf beginnend mit der Erarbeitung einer Migrationsstudie, über die Feinkonzeption bis hin zur Umstellung der

Clients, die momentan[2] noch nicht abgeschlossen ist.

2.1. Erstellung einer Migrationsstudie

Nach einer Ausschreibung durch das Beschaffungsamt des BMI stellten sich die Verantwortlichen der BStU gemeinsam mit der Firma INFORA von Oktober bis Dezember 2002 der Aufgabe, innerhalb von drei Monaten eine Untersuchung durchzuführen. Unter den Gesichtspunkten des fachlich und wirtschaftlich besten Weges und durch Prüfung der strategischen, technischen und monetären Aspekte sollte die beste Lösung für eine Migration der IT-Infrastruktur der BStU gefunden werden.

Diese Untersuchung wurde in vier Abschnitte unterteilt, welche in den folgenden Unterabschnitten kurz erläutert werden:

- Ist-Aufnahme
- Infrastrukturalternativen
- Nutzwertanalyse
- Wirtschaftlichkeitsbetrachtung

Ist-Aufnahme

Die Ist-Aufnahme beinhaltete die Erfassung und die Analyse der bestehenden BStU–spezifischen IT-Infrastruktur und der bestehenden IT-Betriebsorganisation zur Wiedergabe des Ist-Zustands. Dabei wurden durch die Mitarbeiter der INFORA GmbH die gesamte Hard- und Software erfasst und nach ihrer Komplexität und den an sie gestellten Anforderungen in Kategorien eingeteilt, wie z. B. für das Fachverfahren Sachakteneinsicht (Komplexität hoch, Verfügbarkeit muss permanent gewährleistet sein). Weiterhin wurde die Verfügbarkeit an Wissen und Erfahrung bei den Anwendern und den Mitarbeitern des Referates IT/TK aufgenommen und ausgewertet. Durch die Erfassung und Analyse wurden auch Anforderungen zur Weiterentwicklung und Optimierung der bestehenden IT-Infrastruktur aufgestellt.

Infrastrukturalternativen

Ausgehend vom Ist-Zustand wurden vier verschiedene Migrationsalternativen ermittelt (Machbarkeitsstudie). Dabei wurden die Möglichkeiten zur Umsetzung der bestehenden IT-Strukturen auf neue Betriebssystemplattformen untersucht. Die vier ermittelten Infrastrukturalternativen sind in der folgenden Übersicht dargestellt:

1. Reine Microsoft-Umgebung (unter Nutzung der vorhandenen Microsoft-Erfahrung)
2. Aktualisierung der vorhandenen IT-Struktur und Einsatz erster Linux-Server

2 Stand: September 2004

3. Schrittweise Migration auf Linux (unter Nutzung der vorhandenen Novell-Netware-Erfahrung und Verzicht auf Microsoft-Server-Systeme)

4. Migration in einem Schritt zu Linux (harter Schnitt)

Nutzwertanalyse (Qualitative Bewertung)

Es wurde eine vergleichende Bewertung der vier Infrastrukturalternativen nach qualitativen Kriterien auf der Basis des methodischen Mittels Nutzwertanalyse angegangen. Es sollten die zwei besten Infrastrukturalternativen ermittelt und anschließend einer IT-Wirtschaftlichkeitsbetrachtung unterzogen werden.

Das Ergebnis der Analyse zeigte, dass zu den zwei besten Varianten Linux-Netware/Windows XP und Netware-Windows-Linux/Windows XP zählten.

Die Infrastrukturalternative Linux/Linux zeigte trotz ihrer kompletten Open–Source-Ausrichtung für die kurz- und mittelfristige Umstellung der IT-Infrastruktur der BStU deutliche Nachteile. Keines der verwendeten Fachverfahren wäre weiter einsetzbar gewesen, da die meisten mit Visual Basic 6[3] erstellt wurden. Eine Neuerstellung in einer plattformunabhängigen Programmiersprache würde einen längeren Zeitraum in Anspruch nehmen. Weiterhin waren im Bereich der Systemadministratoren und der Anwenderbetreuung nur geringe Kenntnisse im Linux-Bereich vorhanden. Für alle Anwender und Mitarbeiter des Referates IT/TK wären längere Schulungen und Einarbeitungszeiten erforderlich gewesen, die einen wesentlich höheren finanziellen Aufwand zur Folge gehabt hätten.

Die Variante Windows/Windows XP entsprach nicht den strategischen Zielen des BMI zur Ausrichtung der IT-Infrastruktur. Außerdem gab es bei der Bewertung innerhalb der Nutzwertanalyse Nachteile bei der IT-Sicherheit.

Die Variante Netware-Windows-Linux/Windows XP zieht einen wesentlich höheren Bedarf an Hardware nach sich. Für jede Außenstelle wäre z. B. ein neuer MS-Windows-2000-Server erforderlich gewesen. Außerdem hätte man zwei Verzeichnisdienste pflegen müssen. Der Administrationsaufwand für diese Variante wäre wesentlich größer gewesen als bei der Linux-Netware-Variante, in der nur ein Verzeichnisdienst zum Einsatz kommt.

IT-Wirtschaftlichkeitsbetrachtung (IT-WiBe)

Weiterhin wurde zur Nutzwertanalyse, welche eine qualitativ-strategische Bewertung ermöglichte, eine Wirtschaftlichkeitsbetrachtung gemäß der „WiBe 21 – Version 3.0“ der KBSt durchgeführt. Es erfolgte bei zwei Infrastrukturalternativen eine vergleichende Gegenüberstellung bezüglich der quantitativen Kosten und des Nutzens.

Neben der „WiBe 21 – Version 3.0“ wurden auch die „Hinweise und Empfehlungen zur Durchführung von Wirtschaftlichkeitsbetrachtungen bei IT-Update- beziehungsweise Umstellungsvorhaben auf Grundlage der IT-WiBe-97“ der KBSt berücksichtigt.

3 Visual Basic wird unter Linux nicht unterstützt.

Die quantitativ-monetäre Bewertung der Infrastrukturalternativen in der Wirtschaftlichkeitsbetrachtung deckt zusammen mit der qualitativ-strategischen Bewertung der Infrastrukturalternativen in der Nutzwertanalyse all die Bewertungsaspekte, die laut „WiBe 21 – Version 3.0" empfohlen sind, ausreichend ab.

Wie in der Migrationsstudie festgestellt wurde, ist das Ergebnis der Wirtschaftlichkeitsbetrachtung unter dem Aspekt zu werten, dass bei einem solchen umfangreichen Migrationsprojekt nicht unbedingt eingespart werden kann (z. B. beim Personal), was beispielsweise beim Einführen eines neuen Fachverfahrens oftmals möglich ist.

Der Wirtschaftlichkeitsbetrachtung wurde ein Zeitraum von fünf Jahren zu Grunde gelegt. Dabei wurde für die beiden betrachteten Infrastrukturalternativen (Netware–Windows-Linux/Windows XP und Linux-Netware/Windows XP) festgestellt, dass in dieser Zeit haushaltswirksame Kosten anfallen, die durch den haushaltswirksamen Nutzen nicht neutralisiert werden können.

Allerdings ist die BStU aufgrund der technischen Rahmenbedingungen zur Migration der Betriebssystemplattformen gezwungen. Mit einer Migration wird eine strategische und zukunftsorientierte Ausrichtung der IT-Infrastruktur der BStU in Richtung OSS ermöglicht, so dass die notwendigen Investitionen langfristig gerechtfertigt sind. Ferner wird die bestehende IT-Infrastruktur modernisiert, so dass der Funktionsumfang deutlich ausgebaut werden kann.

Zusammenfassung des Ergebnisses der Migrationsstudie

Die Zielsetzung der Studie bestand insbesondere darin, eine sowohl fachliche als auch wirtschaftlich nachvollziehbare Entscheidung hinsichtlich der zukünftigen Ausrichtung der IT-Infrastruktur der BStU zu finden. Ferner sollte unter Berücksichtigung der vorhandenen Ressourcen ein Weg gefunden werden, Open-Source-Software ohne Beeinträchtigung der Arbeitsfähigkeit der Endnutzer einzusetzen (Infora 2002, S. 5):

> „In allen vier Phasen der Migrationsstudie wurde der Anwender (IT–Nutzer) in den Mittelpunkt der Untersuchungen gestellt, da die Bereitstellung einer zuverlässigen, flexiblen und innovativen IT-Infrastruktur zur Bearbeitung der originären Fachaufgaben primäres Ziel des IT-Einsatzes in der BStU ist. Dieser Blickwinkel hatte einen wesentlichen Einfluss auf die Struktur und die Ausrichtung der Migrationsstudie und bildete die Grundlage für das „vierschichtige Schalenmodell"[4] der INFORA GmbH, auf dessen Basis die Untersuchungen strukturiert wurden."

Jeder Untersuchungsabschnitt wurde mit einer Veranstaltung beendet, in welcher der gesamte Ablauf des jeweiligen Abschnitts vorgestellt wurde. Dazu wurden Vertreter der Behördenleitung, der Personalräte, Systembetreuer der Außenstellen und andere Mitarbeiter der Behörde sowie die für die IT der BStU zuständige Mitarbeiterin des

4 Bei diesem Modell steht der Nutzer bzw. der Client im Mittelpunkt der Betrachtung. Um ihn herum werden weitere Schichten bzw. „Schalen" platziert: Services (Fachanwendungen oder spezielle Dienste z. B. für den E-Mail-Zugriff), Infrastrukturdienste (zur Verwaltung der Benutzerdaten oder zur Überwachung des Netzes) und Hardware (Server und weitere Netzwerkkomponenten).

IT-Stabes des BMI eingeladen. Dieses Verfahren wird bis zum heutigen Zeitpunkt für alle Meilensteine des Projektes weiter verfolgt, um eine zeitnahe Informierung aller Entscheidungsträger des Hauses und des BMI zu gewährleisten.

Am Ende der Studie stand ein Ergebnis, das die finanziellen Möglichkeiten der BStU und die Erfahrungen der IT-Mitarbeiter berücksichtigt sowie den strategischen Zielen der Bundesverwaltung entspricht. Das Ergebnis zeigte weiterhin, dass mit der Variante Linux-Netware/Windows XP der Grundstein für den schrittweisen Aufbau einer umfassenden Open-Source-Infrastruktur geschaffen werden kann (Infora 2002, S. 19):

> „Für den Nutzer wird eine „sanfte" (schrittweise) Migration zu Open–Source-Anwendungen möglich gemacht, wobei die strategischen Ziele der Bundesverwaltung weiterhin berücksichtigt und erfüllt werden. Das IT-Personal wird schrittweise auf einen vollständigen Plattformwechsel vorbereitet – vorhandene IT-Qualifikationen und -kompetenzen werden zunächst weitergenutzt. Mit den neuen Systemen können bereits Erfahrungen gesammelt werden."

2.2. Das Jahr 2003: Erstellen der Feinkonzeption und dann: „Testen, testen, testen!"

Mit Abschluss der Migrationsstudie wurde die Entscheidung getroffen, die von der INFORA GmbH vorgeschlagene Infrastrukturalternative Linux-Netware/Windows XP als die zukünftige IT-Infrastruktur der BStU auszuwählen. Bei dieser Alternative kommen Server-seitig die Betriebssysteme Novell 6.5 und SuSE Linux Professional zum Einsatz. Diese Infrastrukturalternative ist somit die erste Stufe der Migration in Richtung Open-Source-Software.

Das Jahr 2003 war durch die Erstellung der Feinkonzeption für die einzelnen Teilschritte der Migration und den Aufbau eines Testnetzes gekennzeichnet.

Um eine reibungslose Migration zu gewährleisten, war es notwendig, in einer Testumgebung alle Bereiche der zukünftigen Infrastruktur ausgiebig zu prüfen. Es wurden aufeinander aufbauende Teilstrukturen (Aufbauphasen) der Infrastruktur erstellt. Um die Gesamtfunktion der Teilstrukturen zu überprüfen, wurden für die jeweiligen Testbereiche Testfälle definiert, durchgeführt und protokolliert. Mit Hilfe dieser Testumgebung konnten frühzeitig Fehler erkannt und behoben werden. Ausfallzeiten während der Server-Migration konnten so verhindert werden. Darüber hinaus wurden alle bei der BStU eingesetzten Software-Pakete der Clients auf MS-Windows-XP-Tauglichkeit überprüft und verschiedene Varianten der Installation und des *Rollouts* „geprobt" und konzeptionell erarbeitet und beschrieben.

Aufgrund der Änderungen an der IT-Infrastruktur (z. B. Anbindung der Außenstellen und Zusammenlegung von Haus- und Zweitnetz) musste das vorhandene IT-Sicherheitskonzept erweitert bzw. überarbeitet werden. Die Änderungen im IT–Sicherheitskonzept zogen weitere Änderungen in diversen Dienst- und Arbeitsanweisungen nach sich. Diskussions- und Anpassungsbedarf gab es vor allem bei den

Browsereinstellungen, der Verwendung aktiver Inhalte, dem Einsatz von Scriptsprachen bei der Neuerstellung der Fachverfahren, der flächendeckenden Nutzung von Internet/E-Mail und den damit verbundenen rechtlichen, kostenmäßigen und administrativen Konsequenzen.

Innerhalb des Feinkonzepts hatten die drei im Folgenden erläuterten Schwerpunkte eine besondere Bedeutung:

- Groupware
- Verzeichnisdienst
- Fachanwendungen

Grundvoraussetzung für die Einführung einer Groupware-Lösung war ebenfalls ein detailliertes Konzept. Da im Laufe des Jahres 2003 die Anforderungen der Anwender an die Funktionalitäten der Groupware immer mehr zunahmen, musste im Rahmen der Tests die Entscheidung zum Einsatz der Groupware revidiert werden. Die geplante Kombination von Novell Netmail mit MS Outlook war den Wünschen der Anwender nicht gewachsen. Daher wurde die Entscheidung getroffen, komplett auf das Produkt Novell GroupWise zu migrieren und auf den Einsatz von MS Outlook auf dem Arbeitsplatz-PC zu verzichten. Da es diese Software inzwischen auch komplett für den Einsatz unter Linux gibt, ist eine Ablösung des Client-Betriebssystems aus Sicht der Groupware problemlos möglich.

Im Verzeichnisdienstkonzept wird die Struktur des Verzeichnisses (*E-Directory*) unter Berücksichtigung der Organisation der BStU sowie der IT-Anforderungen an Netzwerk und Sicherheit in einer hierarchischen Baumstruktur abgebildet.

Im Konzept zur Entwicklung der Fachverfahren wird die Ablösung der Visual–Basic- und 4GL-Programme beschrieben. Zielstellung ist das Angebot Web-basierter Anwendungen über ein gemeinsames Intranetportal der BStU. Dafür wurden zunächst folgende Maßnahmen durchgeführt:

- Schulung der Programmierer in der Programmiersprache Java[5]
- Durchführung von Coachingmaßnahmen zur Einarbeitung und Erstellung eines Testprojektes
- Abschätzen des Zeitaufwandes für die Neuprogrammierung
- Aufbau einer plattformunabhängigen Test-Entwicklungsumgebung[6]

5 Dafür wurde ein externer Dienstleister beauftragt.

6 Diese Umgebung besteht unter anderem aus Eclipse 2.1 mit verschiedenen *Plugins*, dem Jakarta Tomcat 4.1, einem LAMP-Test-Webserver, Informix 9 auf SuSE Linux Professional und dem Java Software Development Kit 1.4. Zukünftig (ab November 2004) wird die Entwicklung mit Eclipse 3 und dem Tomcat 5 erfolgen. Im Zusammenhang mit dieser Grundstruktur wurden weitere verschiedene Open-Source- und kommerzielle Produkte getestet.

2.3. Die Migration im Jahr 2004

Als Zielstellung für die Umstellung der Server-Plattformen wurde definiert, dass der operative Betrieb der Behörde nicht durch migrationsbedingte Einflüsse beeinträchtigt werden darf.

Im Zeitraum vom 6. Januar bis 30. Juni 2004 erfolgten die Server-Umstellung (in der Zentrale der komplette Umstieg auf Novell Netware 6.5) und die Anbindung der Außenstellen an die Berliner Zentrale. Des Weiteren wurden das im Verzeichnisdienstkonzept beschriebene *E-Directory* aufgebaut und alle nötigen Infrastrukturdienste installiert und konfiguriert. Die wichtigsten installierten und konfigurierten Dienste sind in der folgenden Übersicht dargestellt:

- Backup-Server (basierend auf Backup Exec 9)
- ZENworks-Server zur Software-Verteilung
- Anbindung der 13 Außenstellen, einschließlich dem Aufbau einer DMZ (*demilitarized zone*)[7] mit zentralisiertem Webserver, Novell-GroupWise-Server, Applikations- und Datenbankserver
- Konfiguration des Exchange-GroupWise-Connectors und temporäre Synchronisation der Adressbücher von MS Exchange 2000 und Novell GroupWise bis zum Ende der Migration
- Übernahme der Postfächer der Außenstellen von MS Mail 3.0 nach Novell GroupWise
- Datei-Server-Update einschließlich aller Nutzerdaten
- Update des Novell-Netware-Clients und der ersten ZENworks-Komponente auf den Arbeitsplatz-PCs

Neben den technischen Umstellungsprozessen im Jahr 2004 kamen auch andere für die Migration entscheidende Aktivitäten hinzu. Dazu zählen unter anderem die Ausschreibung der Anwenderschulung über das Beschaffungsamt des BMI im Frühjahr 2004 (gewünschter Beginn der Schulung war der 6. September 2004), die Abstimmung zu den Inhalten und dem Aussehen des neuen MS-Windows-XP-Standard-Clients mit MS Office 2003 und dem Novell-GroupWise-Client ab März 2004 und ab Anfang Juli die Beschaffung und Paketierung aller Software-Produkte für das *Rollout* mittels ZENworks.

Die Client-Migration: MS-Windows-XP-Rollout

Am 6. September 2004 begannen die Anwenderschulungen für die Mitarbeiter des Referates IT/TK und für einige Mitarbeiter anderer Referate (so genannte Pilotphase mit ca. 90 Anwendern) bzgl. der Umstellung auf MS Windows XP. Diese Maßnahme sollte zunächst die Abläufe des Umstiegs beim Anwender trainieren und die Möglichkeit geben, eventuelle Probleme noch rechtzeitig zu erkennen und zu beheben.

7 Alle Server in der DMZ sind Linux-Server.

Im Oktober 2004[8] soll dann die eigentliche Schulungs- und Umstellungsphase der Mitarbeiter der BStU beginnen. Am 30. Juni 2005 werden die Arbeitsplätze aller Mitarbeiter der BStU einschließlich der Arbeitsplätze der Mitarbeiter in den Außenstellen auf MS Windows XP umgestellt sein. Die Schulung der Mitarbeiter wird bis dahin entsprechend abgeschlossen sein.

An dieser Stelle sei angemerkt, dass eine Migration zu OSS auf Client-Seite von bestimmten Faktoren abhängig ist. Die Bereitstellung einer leistungsfähigen Software-Verteilung für einen Linux-Client ist eine entscheidende Voraussetzung für die Migration. Zum Beispiel ist die Integration der von Novell aufgekauften Produkte von Ximian in die ZENworks-Suite zur automatisierten Verteilung des Linux-Desktops und der darauf eingesetzten Software notwendig. Weiterhin sind ein leistungsfähiger Browser sowie ein Office-Paket, das kompatibel zu den im Geschäftsbereich und beim Antragsteller eingesetzten Office-Paketen ist, erforderlich.

Die Neuprogrammierung bzw. Überarbeitung der Fachverfahren ist unvermeidlich, was einen hohen Einarbeitungsaufwand und einen größeren Zeitraum als ursprünglich geplant einnehmen wird. Erst mit der Neuprogrammierung wird eine entscheidende Voraussetzung für die Ablösung von MS Windows XP durch Linux auf den Clients geschaffen. Eine 1:1 Umsetzung der bisherigen VB- und 4GL-Programme ist nicht möglich. Das vertraute *Look & Feel* und die Funktionalitäten der Applikationen sollen erhalten bleiben.

Die Migration des Clients hat unmittelbaren Einfluss auf die Arbeitsfähigkeit der Anwender. Die Client-Umstellung erfordert somit eine sehr detaillierte Vorbereitung. Aus Sicht der BStU ist für diesen Schritt eine umfangreiche Schulung jedes Anwenders notwendig.

3. Fazit und Ausblick

Die Infrastrukturalternative Linux-Netware/Windows XP konnte sich sowohl in der qualitativen Bewertung als auch in der Wirtschaftlichkeitsbetrachtung gegenüber den anderen Infrastrukturalternativen behaupten.

Dennoch bringt diese Alternative Nachteile mit sich, wie z. B. den langen Zeitraum bis zur Umsetzung des endgültigen Migrationsziels, die Schulung aller Anwender und die Neuerstellung der Fachverfahren.

Weitere Belastungen für die Anwender ergeben sich durch den Wechsel des Groupware-Produktes (von MS Outlook auf Novell GroupWise) und die Ablösung der Fachverfahren. Der Umgang mit Outlook unterscheidet sich erheblich von GroupWise. Viele über Jahre vertraut gewordene Menüpunkte befinden sich nun an anderen Positionen oder sind anders bezeichnet, einige Funktionalitäten sind in GroupWise nicht enthalten, andere dafür neu hinzugekommen.

Das Upgrade des PC-Betriebssystems und der Office-Version spielt für den Anwender kaum eine Rolle, diese werden nicht als negativ empfunden.

8 Stand: September 2004

Mit dem Abschluss der Schulungs- und Umstellungsmaßnahmen am 30. Juni 2005 werden die in der Migrationsstudie festgelegten Ziele, d. h. die Migration auf Linux–Netware/Windows XP erreicht.

Weiterhin sind Aktivitäten im Bereich Open-Source-Software bis Ende 2005 geplant. Beispielsweise soll das Personalmanagementsystem EPOS für die BStU bis Anfang 2005 auf ein Open-Source-System portiert werden. Bei diesem System soll der Open-Source-basierte Java-Applikationsserver JBoss verwendet und das Informix DBMS genutzt werden. Zusätzlich erfolgt mit der Bereitstellung von Novell Netware 7 (basiert auf Linux-Kernel) das Update aller Novell-Netware-6.5-Server. Die geplante Help-Desk-Lösung soll ebenfalls auf einem Linux-System realisiert werden. Des Weiteren werden momentan Alternativen zur Ablösung des Netzwerkmanagementsystems HP OpenView durch ein Open-Source-Netzwerkmanagementsystem untersucht.

Literatur

Infora (2002), *Managementfassung zur Migratiosstudie der BStU*, Infora GmbH, Berlin. Version 1.3.

Weiterführende Informationen

Detaillierte technische Informationen zu den Software-Komponenten finden sich auf der jeweiligen Herstellerseite:

HP OpenView: http://www.managementsoftware.hp.com/

Cisco Works:
http://www.cisco.com/warp/public/cc/pd/wr2k/prodlit/snms_ov.pdf

Novell Netmail: http://www.novell.com/products/netmail/

Novell GroupWise http://www.novell.com/products/groupwise/

Backup Exec 9:
http://www.veritas.com/Products/www?c=product&refId=139

ZENworks: http://www.novell.com/products/zenworks/

Ximian: http://www.novell.com/linux/ximian.html

Der Webauftritt der Firma INFORA findet sich unter: http://www.infora.de/

Die Schriftstücke der KBSt sind unter den folgenden Adressen abrufbar:

Migrationsleitfaden:
http://www.kbst.bund.de/Anlage304426/Migrationsleitfaden.pdf

SAGA-Dokument:
http://www.kbst.bund.de/Anlage304423/SAGA_Version_2.0.pdf

WiBe 21 – Version 3.0:
http://www.kbst.bund.de/Anlage300441/KBSt-Schriftenreihe+Band+52+%281%2c3+MB%29.pdf

Hinweise und Empfehlungen zur Durchführung von Wirtschaftlichkeitsbetrachtungen bei IT-Update- beziehungsweise Umstellungsvorhaben auf Grundlage der IT-WiBe-97:
http://www.kbst.bund.de/Anlage300627/KBSt-Brief+Nr.+04/2000.pdf

Linux im Rathaus – Ein Migrationsprojekt der Stadt Schwäbisch Hall

HORST BRÄUNER

Im Jahr 2001 traf die Kreisstadt Schwäbisch Hall die Entscheidung, auf Open-Source-Software (OSS) zu migrieren. Zum einen sollten so Kosten gesenkt und zum anderen eine größere Herstellerunabhängigkeit erreicht werden. In einem groben Zeitplan wurde festgelegt, dass sowohl Server als auch Desktops zu migrieren sind. Die besondere Herausforderung lag dabei in der Einführung einer zentralen Administration für Windows- und Linux-Clients und dem Umgang mit der vielfältigen Fachsoftware in den einzelnen Abteilungen. Die zentrale Administration wurde durch ein OpenLDAP-System realisiert und die Fachanwendungen wurden auf zentrale Server portiert, wo sie von den Client-Rechnern über eine spezielle Schnittstelle weiterhin wie gewohnt genutzt werden können.

1. Einleitung

Schwäbisch Hall ist eine große Kreisstadt mit ca. 36 000 Einwohnern, 65 km nordöstlich von Stuttgart. Sie gilt insbesondere wegen der jährlichen Freilichtspiele[1] und der Kunstsammlung Würth als „heimliche" Kulturmetropole. Des Weiteren ist die Kreisstadt aktives Mittelzentrum in der Region Hohenlohe-Franken und regionales Verwaltungs- und Schulzentrum des Wirtschaftsraums Heilbronn-Franken.

Die Verwaltung der Stadt Schwäbisch Hall ist auf mehrere Gebäude im Stadtgebiet verteilt. Vier hauptamtliche Administratoren betreuen derzeit 225 vernetzte PC-Arbeitsplätze, deren Einsatz von einfachen Verwaltungsaufgaben im Kindergarten bis hin zu Geoinformationssystemen in Spezialabteilungen reicht.

Das Organigramm in Abbildung 1 gibt einen genauen Überblick über die Verteilung der PC-Arbeitsplätze in den jeweiligen Bereichen der Verwaltung. Das Rückgrat der IT bildet eine flächendeckende Glasfaser-Verkabelung[2], die bereits in den frühen 90er Jahren eingerichtet wurde. Wie andere Verwaltungen hat die zentral verwaltete EDV der Stadt Schwäbisch Hall eine Historie, die mit Novell Netware/MS DOS begann und über VMS, diverse UNIX-Systeme (OSF/1, SolarisX86 u. a.) und Windows

1 Informationen zu den Freilichtspielen finden sich unter: http://www.freilichtspiele-hall.de/

2 Das Netzwerk arbeitet mit einer Übertragungsgeschwindigkeit von bis zu einem Gigabit pro Sekunde.

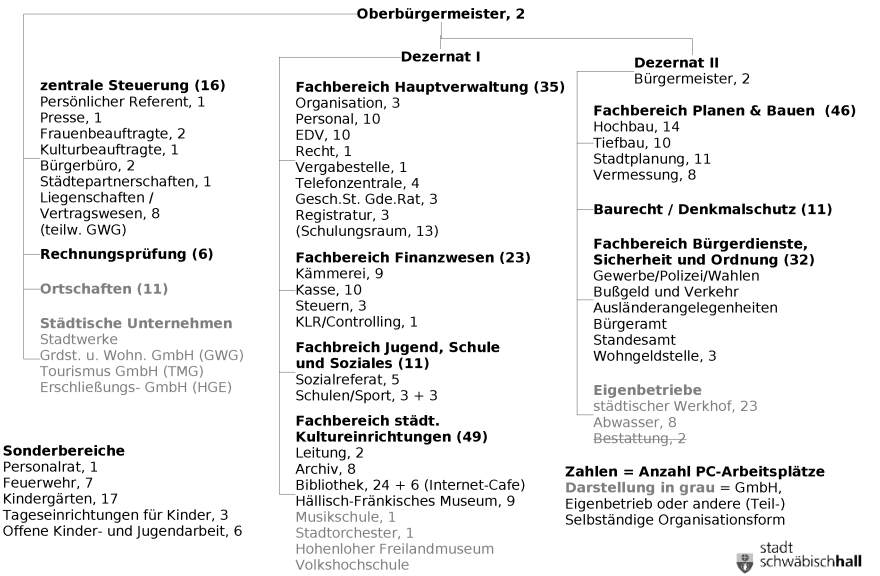


Abbildung 1: Organigramm der Stadtverwaltung Schwäbisch Hall

NT gewachsen ist. Durch berufsbegleitende Fortbildungen, wie z. B. Workshops und Praxisseminare zu verschiedenen informationstechnischen Themen (Netzwerke, Protokolle, Datei-Server, Betriebssysteme etc.) und Zertifizierungen im Bereich Sicherheit (u. a. *Certified Security Engineer*), haben sich die IT-Administratoren eine relativ hohe Kompetenz auch im Linux-/Open-Source-Umfeld erarbeitet.[3]

Im Jahr 2001 wurde von Microsoft angekündigt, die Unterstützung für das zu dieser Zeit in Schwäbisch Hall eingesetzte Produkt Windows NT 4.0 einzustellen. Wann dies auch für das genutzte MS Office 2000 erfolgen sollte, war nur eine Frage der Zeit, da die Nachfolgeprodukte MS Windows 2000/XP und MS Office XP bereits angekündigt und aktiv beworben wurden. Es mussten also Überlegungen angestellt werden, ob Alternativen vorhanden sind, welches Budget für eine mögliche Migration zur Verfügung steht und wann und wie diese durchgeführt werden kann.

Dazu kam, dass die Stadtverwaltung Schwäbisch Hall im selben Jahr einen Steuereinbruch von 85% des Gewerbesteueraufkommens hinnehmen musste. Der daraufhin aufgestellte Plan zur Bewältigung dieser Finanzkrise sah sowohl kurzfristige Einsparungen als auch die langfristige Reduzierung der Ausgaben vor. Die EDV-Abteilung war, ebenso wie sämtliche andere Abteilungen, aufgefordert, Vorschläge aus ihrem Bereich anzubieten.

3 Zudem bildet die Stadtverwaltung regelmäßig Fachinformatiker (Fachrichtung Systemintegration) aus, denen dieses Wissen qualifiziert vermittelt wird. Die „neuen" Themen Open Source (OS) und Linux sind selbstverständlich Bestandteil der Ausbildung.

Deren Vorschlag führte letztlich zu dem Projekt „Linux im Rathaus“, mit dem Ziel, die Verwaltung auf Open-Source-Software umzustellen. Die Aspekte Unabhängigkeit, Sicherheit, Einsparung und Förderung des Wettbewerbs bildeten die Kernpunkte des Vorschlags. Die jeweilige Betrachtung dieser Aspekte belegt die relativen Vorteile einer Migration auf OSS:

Unabhängigkeit:

Der auslaufende Support für Microsoft-Produkte zwang die Verwaltung zu einer wie auch immer gearteten Migration. Eine Migration innerhalb der Microsoft-Welt hätte zu Lizenzkosten von 250 000 € in den folgenden zwei Jahren geführt.[4] Das Problem des eingestellten Supports hätte sich allerdings durch das Fortbestehen der Abhängigkeit von einem Hersteller jederzeit wiederholen können.

Hinzu kam, dass die Hardwareausstattung der Stadtverwaltung zuletzt 1997/98 großflächig erneuert wurde – anschließend nur punktuell. Damit stellten die gestiegenen Hardwareanforderungen der angekündigten neuen Versionen der Microsoft-Produkte ein weiteres Problem für die bestehende Hardware dar.

Eine weitere Herausforderung waren die in der Verwaltung eingesetzten speziellen Fachverfahren, die teilweise seit den 70er Jahren genutzt wurden und keine oder nur bedingt Schnittstellen zu den aktuellen Microsoft-Betriebssystemen bzw. MS-Office-Komponenten hatten.

Sicherheit:

Neben den zunehmenden Problemen mit Angriffen durch Viren, Trojaner und Würmer gibt es drei Aspekte, die das Thema Sicherheit in einer öffentlichen Verwaltung ausmachen:

- Vertraulichkeit von personenbezogenen Daten:
  Es muss ein Schutz der Daten vor dem Zugriff durch unbefugte Personen gewährleistet sein.
- Integrität von personenbezogenen Daten:
  Es dürfen keine unerwünschten Veränderungen an Daten stattfinden. Alle Änderungen an den Daten müssen daher nachvollziehbar protokolliert werden.
- Verfügbarkeit:
  Eine hohe Verfügbarkeit der IT-Systeme ist für das Funktionieren der Verwaltung unerlässlich.

Software-Komponenten, die solche Anforderungen erfüllen, sollten bereits Grundbestandteil der in der öffentlichen Verwaltung eingesetzten Betriebssysteme sein und nicht über kostenpflichtige Lizenzen und Software von Drittanbietern separat nachgerüstet werden müssen. Wenn zudem der Bereich Datenschutz betrachtet wird, stellt ein unabhängig überprüfbarer Quelltext der

4 Stand: 2002, MS-Select-Vertrag (Bestandteil eines speziellen Volumenlizenzprogramms)

eingesetzten Software einen sehr vorteilhaften Aspekt bei der Beschaffung dar. Eine Verwaltung wird dadurch in die Lage versetzt, gegenüber ihren „Kunden" Datensicherheit zu beweisen.[5]

Einsparungen:

Dass Einsparungen möglich sind, belegt der Einsatz von Linux bzw. Open-Source-Software (OSS) auf den Verwaltungs-Servern seit 1997. Damals aus der Not geboren, wurde hier Linux als Basissystem eingeführt. Es gab unter der proprietären Microsoft-Umgebung keinen DHCP-Server, keinen gut administrierbaren DNS-Server und keinen E-Mail-Server, der ausschließlich Relay-Funktionen[6] besitzt. Nach und nach wurden weitere Dienste, wie Samba (Datei-Server), CUPS (Common Unix Printing System – ein Druck-Server) und SQL-Datenbanken, auf dieselbe Hardware konsolidiert.

Im Ergebnis wurde festgestellt, dass die Hardware seit 1997 nicht getauscht werden musste, Software kostenlos zur Verfügung stand und wenig administrativer Aufwand für Linux betrieben werden musste. Extrapoliert man diesen Effekt vom Server auf die Vielzahl von Clients, müsste der Einspareffekt dort um ein Vielfaches höher sein. Rechnet man dazu noch die Einsparpotentiale auf der Client-Seite hoch, so kommt man zu einem Volumen von ca. 500 € pro Arbeitsplatz allein an Lizenzkosten beim Verzicht auf proprietäre Software. Bei den Schulungskosten wurde davon ausgegangen, dass der Aufwand innerhalb der Microsoft-Produktlinie (Erfahrung aus vergangenen Migrationen, z. B. von MS Windows 3.11 zu MS Windows NT oder von MS Office 95 zu MS Office 2000) mindestens so hoch ist wie bei einem Wechsel zu OSS. Deshalb blieben sie für die Kalkulation einer möglichen Einsparung unberücksichtigt.

Wettbewerb:

Normalerweise erwirbt eine Verwaltung keine Einzelkomponenten und fügt sie zu einer Gesamtlösung zusammen, sondern bedient sich eines Dienstleisters, der eine maßgeschneiderte Lösung anbietet. Wenn dieser meist lokal oder regional angesiedelte Dienstleister proprietäre Komponenten bei seiner Lösung einsetzt, muss er einen Teil seines Preises für die Lösung in Form von Lizenzgebühren an den Lizenzgeber abführen oder die Lizenzgebühren bei seiner

5 Am Beispiel eines Eingabe-Verarbeitung-Ausgabe-Modells (EVA-Modell) wird der Unterschied deutlich: Man nehme als Eingabe Kundendaten, wie z. B. Name, Vorname, Augenfarbe. Die Verarbeitung entspricht der elektronischen Kombination dieser Daten in einem Datensatz. Die Ausgabe könnte eine Plastikkarte mit diesen Daten in maschinenlesbarer Form (Personalausweis) sein. Da bei einem proprietären System der Verarbeitungsteil geschlossen (d. h. die Software liegt nur im binären, also nur vom Computer und nicht vom Menschen lesbaren, Code vor) ist, kann nicht überprüft werden, was zu welchem Zeitpunkt mit den Daten geschieht und wie sie verarbeitet (z. B. ob der Datensatz per Internet auf die Fidschi-Inseln kopiert wird oder in der Datensammlung eines Geheimdienstes landet) werden, um aus der Eingabe die Ausgabe zu erzeugen. Der Sicherheitsfaktor besteht nun darin, dass bei einem Open-Source-System der Verarbeitungsteil öffentlich (d. h. der Code der Software liegt auch als Quelltext vor, der für den Menschen nachvollziehbar ist) ist und daher können solche Manipulationen (oder Nebenwirkungen) dokumentiert ausgeschlossen werden.

6 Das heißt eine E-Mail von A nach B zu befördern.

Preiskalkulation berücksichtigen bzw. aufschlagen. Setzt ein Dienstleister dagegen OSS ein, so sind grundsätzlich keine Lizenzgebühren fällig, und die Lösung kann entweder kostengünstiger für die Verwaltung angeboten werden oder es wird ein höherer Gewinn aus der Lösung erzielt. Für die Verwaltung ergeben sich dann bei einem lokalen oder regionalen Dienstleister die Vorteile niedrigerer Ausgaben und eines *Return of Invest* durch die mögliche Einnahme von Gewerbesteuern. Damit kann die Unterstützung von OSS für eine Kommune durchaus auch als Mittel zur Wirtschaftsförderung gesehen werden.

Die Migrationsentscheidung fiel also nach Erwägung der beschriebenen Punkte deutlich für OSS aus. Nachdem die Inhalte der genannten Kernpunkte klar beschrieben waren, konnte das Projekt beginnen. Der folgende Abschnitt gibt den Projektverlauf wieder.

2. Projektverlauf

Entscheidungen im kommunalen Umfeld sind nicht immer an große Formalismen gebunden. So führten die genannten Motivationen bei einer Besprechung mit dem Oberbürgermeister der Stadt Schwäbisch Hall, Hermann-Josef Pelgrim, zum Startschuss für die Migration.

2.1. Vorab gestellte Bedingungen

Die Migration sollte die folgenden Bedingungen erfüllen:

- Die verwendete Fachsoftware für Verwaltungsprozesse muss „laufen".
- Büro-Software, also der klassische Office-Bereich (mit Textverarbeitung, Tabellenkalkulation, Präsentation usw.), muss „kompatibel" sein, d. h. es darf kein Informationsverlust beim Datenaustausch auftreten.
- Die Umstellung hat neben der gewohnten Arbeit zu erfolgen, d. h. durch die Umstellung selbst darf grundsätzlich kein höherer Aufwand für die Arbeiten des normalen Tagesablaufs entstehen.
- Das Ergebnis soll eine vertraute Arbeitsumgebung sein, damit ein minimaler Schulungsaufwand für die Mitarbeiter entsteht.
- Als Zugabe soll ein gemeinsamer (Gruppen-) Terminkalender eingeführt werden.

Die Absicht, eine Linux-/OSS-Migration durchzuführen wurde auf der Linuxworld-Expo 2001 in Frankfurt im Rahmen einer Podiumsdiskussion unter Leitung der Zeitschrift „Computerwoche" vorgestellt. Die Firmen IBM und SuSE haben sich als Partner angeboten, dieses Vorhaben als Pilotprojekt zu unterstützen.

2.2. Zeitplan

Im Dezember 2001 wurde ein Zeitplan aufgestellt, der vorsieht, dass die Umstellung bzw. das Projekt „Linux im Rathaus“ Ende 2004 abgeschlossen wird. Im Vergleich zu standardisierten Prozessen war und ist eine detaillierte Zeitskala hier praktisch unmöglich, da eine solche Umstellung nach dem Wissensstand der Verantwortlichen noch nie erfolgte. Daher wurden nur zwei Abschnitte festgelegt:

1. Server umstellen
2. Desktops abteilungsweise umstellen

Alle weiteren Untergliederungen sollten der Situation entsprechend erfolgen. Die in Schwäbisch Hall gesammelten Erfahrungen können dann bei weiteren Projekten dieser Art zu einem „echten“ Projektplan genutzt werden. Zudem wurde durch die nur grobe Zeitplanung eine größtmögliche Flexibilität gewahrt.

2.3. Umzustellende Bereiche

Erster Schritt bei der Umstellung war die Einführung einer zentralen Administration, sowohl für MS-Windows-Clients als auch für die zunehmende Anzahl der Linux–Clients. Realisiert wurde dies mit OpenLDAP (*Lightweight Directory Access Protocol*) im Rahmen der Abschlussarbeit zur Fachinformatikerin als zentralem Verzeichnis aller zu verwaltender Objekte (vgl. Stichler 2002), wie z. B. Benutzer, Freigaben, E-Mail etc. Der Verzeichnisdienst wurde mit dem Datei-Server „Samba“ kombiniert, um den MS-Windows-Clients einen *Primary Domain Controller* (PDC) als Server für die Anmeldung zur Verfügung zu stellen. Linux-Clients können sich direkt gegen den LDAP-Server authentisieren. Die Abbildung 2 stellt diesen Sachverhalt grafisch dar. Die Infrastruktur wurde den Forderungen nach Hochverfügbarkeit angepasst und besteht nun aus einem Quartett von jeweils vier IBM-x-Series-Servern, die als Cluster alle notwendigen Dienste zur Verfügung stellen. An Software kommt *SuSE Linux OpenExchange Server* (SLOX) und *SuSE Linux Enterprise Server* (SLES) zum Einsatz. Diese Server decken die zentralen Bereiche Benutzerverwaltung, Datei-Server, Druck-Server, E-Mail und Gruppenkalender ab.

Die Umstellung aller Arbeitsplätze bezüglich Office-Software wurde parallel zur Einführung der Groupware vorgenommen. Auf allen Arbeitsplätzen (auch den MS–Windows-PCs) wird das Open-Source-Produkt OpenOffice eingesetzt. Für die Übergangsphase wurde OpenOffice auf den MS-Windows-Arbeitsplätzen parallel installiert.

Als Alternative zu typischerweise verteilten[7] MS-Access-„Datenbanken“ wurden die Daten (Tabellen) auf ein zentrales MySQL-System portiert und mit Web-basierten Benutzeroberflächen ausgestattet bzw. die Tabellen über ODBC direkt in OpenOffice (z. B. Textverarbeitung) integriert. Ausnahmsweise wurde MS-Access unter *CrossOver Office*, einem kommerziellen Windows-Emulator speziell für die MS-Office-Packete, eingesetzt.

7 Verteilte Daten bedeuten, gegenüber zentral gespeicherten, höheren Verwaltungsaufwand.

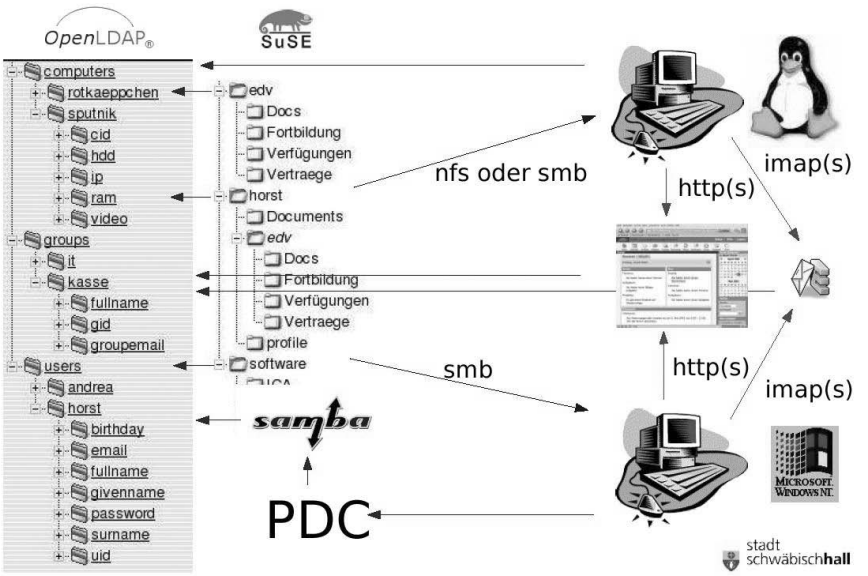


Abbildung 2: Umstellung zentrale Verwaltung

Für eine gemischte Umgebung aus MS-Windows- und Linux-Desktops kam von Anfang an nur eine Web-basierte E-Mail-/Kalender-Lösung in Frage. Mit dem Partner SuSE bot sich an, den SLOX einzusetzen, der die gewünschten Funktionalitäten Gruppenkalender und E-Mail-Client abdeckt und darüber hinaus Optionen bis hin zum *Workflow* enthält. Derzeit werden ausschließlich Kalender und E-Mail genutzt.

2.4. Besonderheiten des Projekts

Zu den Besonderheiten des Projekts zählen insbesondere die bereits vorhandene Fachsoftware und der neu eingeführte universelle Client, die im Folgenden vorgestellt werden.

Fachsoftware

Verwaltungen sehen oft die größte Herausforderung bei einer Migration zu OSS in der Verfügbarkeit der vorhandenen Fachsoftware für ihre Verwaltungsprozesse. Hier muss unterschieden werden, auf welche Weise eine Verwaltung Fachsoftware einsetzt. In Baden-Württemberg z. B. werden als Ergebnis eines Konzepts aus den 70er Jahren größere Fachverfahren, wie z. B. „Einwohnermeldeamt“, von kommunalen Rechenzentren zur Verfügung gestellt. Diese Verfahren laufen meist auf Großrechnern (IBM z-Series) oder anderen Hosts (z. B. BS2000). Typische PC-Verfahren werden von den Rechenzentren auf zentralen Citrix-Farmen betrieben.

Fachverfahren werden den angeschlossenen Kommunen im ASP-Modell (*Application Service Providing*) angeboten, d. h. die Kommune betreibt am Arbeitsplatz lediglich eine Emulations-Software und das Verfahren an sich wird im Rechenzentrum betrieben.

„Klein"-Verfahren sind Programme, die auf wenigen Arbeitsplätzen betrieben werden oder Programme, von denen mehrere an einem Arbeitsplatz installiert sind und die für sehr spezifische Aufgaben, wie z. B. „Fundbüro", eingesetzt werden. Sie sind lokal installiert. Diese Verfahren betreibt jede Kommune für sich auf den Desktops, wobei die Rechenzentren hier Empfehlungen für diese Programme aussprechen.

In Schwäbisch Hall werden insgesamt 84 verschiedene Fachverfahren betrieben – die Mehrzahl davon als Einzelversionen auf den Desktops bei wenigen Anwendern pro Verfahren. Die zugehörige Fachsoftware lässt sich wie folgt kategorisieren:

– Fachsoftware im ASP-Modell

  Verfahren, die im ASP-Modell angeboten werden, sind in der Regel unproblematisch, da die typischen Client-Komponenten auch unter Linux zur Verfügung stehen, wie x3270 (ein IBM-Terminalemulator für das X-Window-System) um Zugriff auf Host-Umgebungen, Clients zum Zugriff auf Citrix-Server und SAP-Clients.

– Fachsoftware im Client-Server-Modell

  Bei den Client-Server-Verfahren, wie z. B. Ausschreibung im Baubereich (AVA), die die Kommune Schwäbisch Hall lokal einsetzt, ist die Stadt Schwäbisch Hall in der glücklichen Lage, dass bis auf das „Standesamtsprogramm" des Fachverlags für Standesamtswesen, für fast alle Fachverfahren Alternativen unter Linux angeboten werden.

– Fachsoftware Desktop

  Die Vielzahl der Klein- und Kleinstverfahren macht es schwierig, immer sofort Alternativen für Linux zu finden. Es gibt eine Reihe von Möglichkeiten, typische MS-Windows-Programme auch unter Linux zu betreiben, wie z. B. *Wine*, *rdesktop*, *CrossOver Office* oder *VMWare*. Nicht alle Applikationen laufen allerdings in solchen Emulationen und teilweise müssen die Emulatoren für jede Applikation anders angepasst werden (typischerweise „Wine"). Der Ansatz in Schwäbisch Hall ist daher, die Verfahren, sofern sie unverzichtbar sind, auf einen MS-Terminalserver zu migrieren, um den administrativen Aufwand zu reduzieren und einen universellen Client auf den Desktops einzusetzen.

Universeller Client

In einer gemischten MS-Windows-/Linux-Umgebung ist es nahezu unmöglich, dieselben Client-basierten Programme für beide „Welten" zu finden. Ausnahmen bilden hier bisher nur Java-Entwicklungen und teilweise Web-basierte Verfahren. Die Stadtverwaltung geht daher weg vom Client hin zum Server. Da allerdings absehbar ist, dass Programme, die heute noch unter MS Windows laufen, zukünftig unter Linux

betrieben werden, müsste grundsätzlich beim Wechsel der Applikation der Client neu konfiguriert werden. Dieser administrative Aufwand kann durch Einsatz des Open–Source-basierten Terminalservers „NoMachine“ vermieden werden, da die Client–Komponente für nahezu alle Plattformen verfügbar ist (Windows, Linux, MacOS etc.).

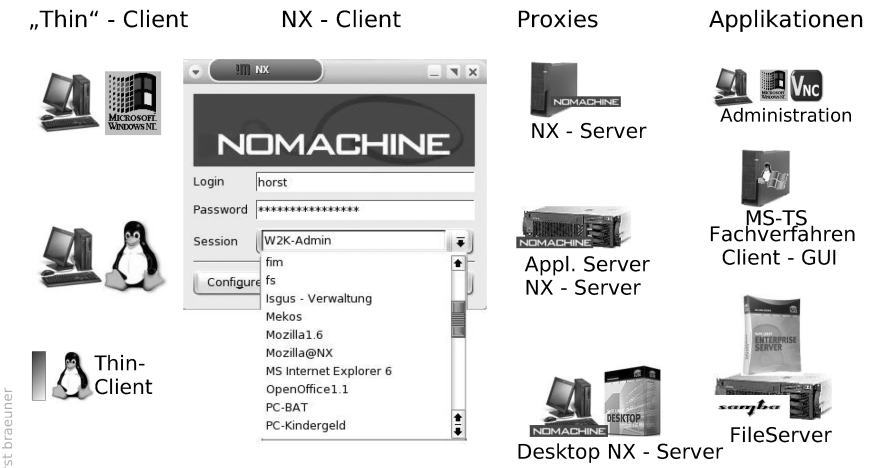


Abbildung 3: Universeller Client für Fachsoftware

Genutzt wird der universelle Client in zwei Anwendungsbereichen:

– Fachsoftware:

  Für die Fachsoftware wird ein Portal etabliert, das gegenüber dem Client die Darstellung übernimmt und ein standardisiertes Protokoll verwendet (komprimiertes, mittels *Secure Socket Layer* verschlüsseltes X11-Protokoll). Der Client verbindet sich zum Portal. Ob die Fachapplikation nun von einem MS–Windows-Server oder von einem UNIX-Server zur Verfügung gestellt wird, ist für den Client transparent. Bei einer Migration einer bisherigen MS-Windows-Applikation auf einen Linux-Server wird dem Client also automatisch die richtige Applikation angeboten. Abbildung 3 zeigt den universellen Client im Zusammenhang mit der Fachsoftware.

– *Local Area Network*, *Wide Area Network* bzw. *Mobile*:

  Der verwendete „NoMachine“-Client ist so performant, dass selbst bei schmalbandigen Verbindungen, z. B. über *General Packet Radio Services* (GPRS), Fachapplikationen weitergeleitet werden können. Das bedeutet, dass die Verwaltung nicht mehr auf eine bestimmte Infrastruktur angewiesen ist und ihre Dienste grundsätzlich überall anbieten kann.

Weitere Informationen zum Einsatz von NoMachine finden sich im Abschnitt 2.8. „Neu evaluierte Projekte“ (Unterabschnitt: „Thin Client“).

2.5. Umstellungsprozess

Der Umstellungsprozess musste mit minimalen Kosten erfolgen. Damit war klar, dass grundsätzlich keine neuen Desktops angeschafft werden konnten. Für die Realisierung wurden daher anfangs 40 PCs beschafft, die als „Verschiebemasse“ die ersten Umstellungen zuließen. Die aus der Fachabteilung abgebauten MS-Windows-Desktops wurden in einem technischen Review auf ihre zukünftige Verwendbarkeit überprüft. Abschließend wurde der SuSE Linux Desktop (SLD) installiert. Bei der Umstellung der nächsten Abteilung wurden die Desktops wiederum „recycelt“.

Begleitend zu der technischen Migration wurden in Gruppen von acht bis zwölf Mitarbeitern die Schulungstermine abgestimmt, wobei die Abstimmung an sich den längsten Zeitfaktor ausmachte. Die Schulungen bestanden aus drei- bis vierstündigen Modulen, die während der Arbeitszeit absolviert wurden. Sie deckten u. a. die Themen „Umgang mit dem Desktop“ und „Unterschiede zwischen MS Office und OpenOffice“ ab. Während die Mitarbeiter geschult wurden, wurden die Arbeitsplätze getauscht, sodass nach erfolgreich absolvierten ein bis zwei Schulungstagen die neuen Desktops produktiv genutzt werden konnten. Für die Nachbereitung standen die Mitarbeiter der EDV über eine Hotline zur Verfügung. Sämtliche Fragen und Probleme werden in einer Datenbank erfasst (*Trouble Ticket System*), damit der Erfolg der Schulung gemessen und bei wiederholten Problemen der Schulungsinhalt ergänzt oder überarbeitet werden kann.

2.6. Projektbegleitende Erfahrungen und Aktivitäten

Es wurden keine Leistungsverträge mit den Fachabteilungen definiert oder vereinbart: ein Problem, das die IT der Stadt Schwäbisch Hall von Beginn des Projektes an begleitete. Es entstanden Reibungseffekte dadurch, dass die Fachabteilungen nicht wussten, was die EDV zu leisten in der Lage war und somit auch keine definierten Anforderungen an die EDV stellen konnten. Die Ansprüche der Fachabteilung überstiegen oft die Kapazitäten der vorhandenen EDV-Abteilung. Eine weitere „vielleicht in Zukunft problematische“ Erfahrung des Projektes ist, dass der Bereich Organisation von der IT abgekoppelt betrieben wird, sodass organisatorische Verbesserungen nicht Bestandteil des Umstellungsprozesses nach OSS sein konnten.

Andererseits gab es auch positive Aspekte: Die große öffentliche Wirkung, die Schwäbisch Hall durch Publikation der Entscheidung erreicht hat, hatte dazu geführt, dass sehr schnell das Interesse der kommunalen Rechenzentren erregt wurde und hier eine Mitarbeit angedacht wurde. Weiterhin folgten Partnerschaften mit der Industrie und dem Linux-Verband e. V. (LIVE).

Der Linux-Verband hat auf Grund der persönlichen Mitgliedschaft des Autors den Arbeitskreis „Public Sector“ mit dem Autor als Vorsitzenden ins Leben gerufen. Der Arbeitskreis hat vorgeschlagen, dass im Rahmen einer Magisterarbeit eine

„Datenbank" erstellt wird, in der die Prozesse einer öffentlichen Kommunalverwaltung mit den derzeit eingesetzten proprietären Programmen beschrieben werden. Die Community bzw. Dienstleister aus dem OSS-Umfeld haben damit die Möglichkeit, eigene oder in Kooperation mit den derzeitigen proprietären Anbietern Entwicklungen zu Fachverfahren anzubieten. Die Finanzierung dieser Arbeit ist derzeit noch nicht geklärt.

Um weiteren Kommunen den Umstieg auf OSS zu erleichtern, hat die Hallkom Service und IT GmbH (eine Tochter der Stadtwerke Schwäbisch Hall GmbH) mit der Topalis AG 2004 die Firma Tosipa SHA GmbH gegründet, welche die Erfahrungen und Ergebnisse aus dem Projekt der Stadt Schwäbisch Hall als Dienstleistung anbietet.

2.7. Widerstände und Erfolge

Von vornherein standen einige Mitarbeiter dem Projekt skeptisch gegenüber. Unkenrufe und die teilweise anfangs von der Presse verbreiteten eher kritischen Meinungen zu OSS waren wohl der Grund für ein gewisses „Unwohlsein". Um alle beschäftigten Mitarbeiter der Verwaltung von Beginn an in die Thematik einzuführen, wurde entsprechende Aufklärungsarbeit geleistet.

Widerständen wurde auch durch die Verteilung von CDs mit aktueller OSS an alle interessierten Mitarbeiter vorgebeugt. Inhalt sind OpenOffice, der Webbrowser Mozilla und das Grafikprogramm GIMP, jeweils auch in der MS-Windows-Version. Damit kann der Mitarbeiter OSS auch auf seinem privaten PC nutzen, ohne sie extra aus dem Internet herunterladen zu müssen.

2.8. Neu evaluierte Projekte

Im Rahmen des Migrationsprozesses wurden auch neue Projekte evaluiert und auf ihre Realisierbarkeit hin geprüft. Als kreative Beispiele seien die folgenden drei Projekte vorgestellt:

– „Bio-SignOn" (angeregt von IBM)

  Beim „Bio-SignOn" geht es darum, biologische Merkmale zur Authentisierung der Nutzer einzuführen. Neu daran ist, dass nicht Fingerabdruck oder Iris-Erkennung die biologische Komponente bilden, sondern die eigene Unterschrift, wobei der Schriftzug um die eindeutigen Merkmale Geschwindigkeit, Druckstärke, Ausschläge usw. ergänzt wird. Als Ergebnis wurde festgehalten, dass die Methode grundsätzlich für eine eindeutige Identifizierung geeignet ist. Die Handhabung ist allerdings in der Praxis noch nicht ausgereift. Insbesondere das Unterschreiben auf einem elektronischen Tablett ist für „normale" Benutzer sehr ungewohnt und bedarf längerer Übung.

– „Mobiles Rathaus" (entstanden während der Neugliederung der Stadtverwaltung 2003)

  Das Projekt „Mobiles Rathaus", derzeit im Praxistest, soll zu weiteren Einsparungen führen. Die Stadt Schwäbisch Hall betreibt mehrere Außenstellen in

Teilorten, um vor Ort Bürgerservice zu leisten. Derzeit können so Behördengänge in den Teilorten erledigt werden. Dem Angebot sind allerdings Grenzen gesetzt: Es kann wegen der Vielfalt der möglichen Dienstleistungen und den begrenzten Personalressourcen nicht die komplette Palette der möglichen Bürgerdienste angeboten werden. Außerdem sind die Kontaktzeiten eingeschränkt. Diese Außenstellen werden in festen Immobilien (Bezirksrathäusern) betrieben. Der Unterhalt dieser, zum Teil nur stundenweise belegten, Gebäude ist ein Kostenfaktor. Wenn es gelingt, die Dienstleistung unabhängig von diesen Rathäusern anzubieten und darüber hinaus den Katalog der Dienstleistungen zu erweitern oder zu vervollständigen, wäre eine anderweitige Nutzung der Gebäude möglich und könnte, z. B. bei Vermietung als Gewerbe- oder Wohnobjekt, zu zusätzlichen Einnahmen führen. Zweck dieses Projekts ist es zu testen, wie viele Angebote unabhängig von den Bezirksrathäusern realisierbar sind. Es kommen als „Ersatz" z. B. Kindergärten, Schulen oder Bankfilialen in Betracht. Erstere sind in kommunalem Besitz und werden ebenfalls nur zeitweise genutzt. Leistungen in Filialen Dritter zu verlegen, hat die Deutsche Post AG in Kombination mit Lebensmittelgeschäften vorgemacht. Es ist auch eine komplett mobile Variante im Praxistest. Hier erfolgt der Zugriff auf die Verwaltung über die vorhandenen Funknetze (D1, D2, E-Plus und O2) und mit Hilfe eines „NoMachine"-Clients. Des Weiteren bieten sich Möglichkeiten bis hin zum kostenpflichtigen „Home-Service", wie ihn andere Dienstleister (z. B. Versicherungen) bereits im Außendienst eingerichtet haben, an.

– „Thin Client" (als Ergänzung bzw. Erweiterung zu „NoMachine")

  Die Erfahrung in diesem Projekt, insbesondere der entfernte Zugriff auf die Verwaltung, hat zu dem weiteren Projekt „Thin Client" geführt. Das heißt der eingesetzte (mobile) Arbeitsplatz wird auf die Funktionen Verbindung zur Verwaltung und Darstellung einer grafischen Benutzeroberfläche reduziert. Zu den Einsparungsmöglichkeiten durch den Wegfall der Immobilien kommen dann noch Einsparungen bei der Hardware für den Arbeitsplatz, der Administration und den Betriebskosten. Sollte dieses Projekt erfolgreich sein, wird wiederum als Folge davon der Einsatz von „Thin Clients" in der Kernverwaltung geprüft.

3. Momentaner Status und Fazit

Die eigentliche Umstellung der Arbeitsplätze wurde im Sommer 2003 begonnen. Die erste „Abteilung" nach der IT, waren der Oberbürgermeister und dessen engere Umgebung. Es folgten Stadtbibliothek, Rechnungsprüfung, Bürgerbeauftragter, Kämmerei, Hauptverwaltung usw. Den Abschluss bildete der technische Fachbereich. Die dezentralen Arbeitsplätze, z. B. in Kindergärten, werden nach Bestandsaufnahme am Ende des Umstellungsprozesses entweder alle auf einmal umgestellt oder beim Ersatz der Hardware. Die Umstellung der städtischen Tochterunternehmen erfolgt auf Wunsch des jeweiligen Geschäftsführers. Bisher hat sich bereits die Touristik- und

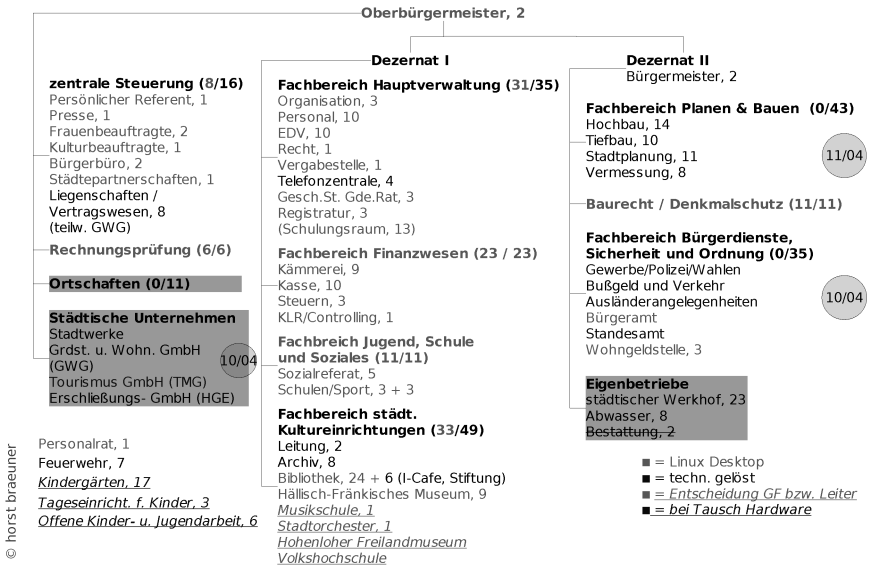


Abbildung 4: Status der Umstellung (Stand: 2004)

Marketing GmbH dazu entschlossen umzusteigen. In Abbildung 4 ist der Status der Migration zusammengefasst dargestellt.

Momentan (Stand: Oktober 2004) sind alle zentralen Bereiche der Verwaltung umgestellt. Bis Ende Oktober 2004 werden auch noch die technischen Abteilungen migriert. Bis Ende des Jahres soll das Projekt abgeschlossen sein. Das Fazit des Projektes lässt sich mit den Worten des Bundesinnenministers Otto Schily aus dem Rahmenvertrag mit IBM am ehesten beschreiben (Bundesministerium des Innern 2002):

> „Wir erhöhen die IT-Sicherheit durch die Vermeidung von Monokulturen; wir verringern die Abhängigkeiten von einzelnen Software-Anbietern, und wir sparen beim Kauf der Software und bei den laufenden Kosten. Damit sind wir Vorreiter, eine größere Vielfalt in der IT-Landschaft zu schaffen."

Literatur

Bundesministerium des Innern (2002), 'Kooperationsvertrag mit IBM über den Einsatz von Open-Source-Produkten: Schily öffnet die öffentliche Verwaltung für Linux', Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnologie in der Bundesverwaltung im Bundesministerium des Innern, http://www.kbst.bund.de/Anlage301643/Pressemitteilung+zum+Kooperationsvertrag+BMI+mit+IBM.pdf [22. Jan 2005]. Presseerklärung des BMI vom 3.6.2002.

Stichler, S. (2002), 'Zentrale Verwaltung einer gemischten MS-Windows-/Linux-Umgebung', Stadt Schwäbisch Hall, intern.

Weiterführende Informationen

Detaillierte Informationen zu der Emulator-Software und zu den Terminalserver–Lösungen finden sich auf der jeweiligen Projekt- bzw. Herstellerseite:

Wine: http://www.winehq.com/

rdesktop: http://www.rdesktop.org/

CrossOver Office: http://www.codeweavers.com/

VMWare: http://www.vmware.com/

NoMachine: http://www.nomachine.com/

Der Webauftritt des Linux-Verband e. V. ist unter der Adresse http://www.linux-verband.de/start.html abrufbar.

Migration auf Samba/OpenLDAP bei der Norddeutschen Affinerie AG

JÖRG MEYER UND CARSTEN BRUNKE

Im März 2003 stand die Norddeutsche Affinerie AG (NA) vor der Entscheidung, möglicherweise wichtige Teile der IT-Infrastruktur von proprietären Produkten auf freie Software umzustellen. Die NA ist Europas größter Kupferhersteller und weltgrößter Kupferrecycler, die IT-Leitung verantwortet an den bundesdeutschen Standorten den Betrieb für ca. eintausend Nutzer. Nach einer intensiven Planungs- und Evaluationsphase fiel im August 2003 der Startschuss für die Migration der Datei- und Verzeichnisdienste auf Samba und OpenLDAP in Zusammenarbeit mit einem externen strategischen Partner, der inmedias.it GmbH. Seit Ostern 2004 ist die neue Struktur live und erreicht eine Verfügbarkeit von 99,7%, basierend auf einem Linux-Server mit Standardhardware. Statt eingeplanter neunzig Tage *Troubleshooting*[1] im Anschluss an die Migration waren nur fünf Tage notwendig, bis ein „geräuschloser" Tagesbetrieb erreicht war. Nicht nur die Anwender profitieren von der gewonnenen Souveränität der Konzern-IT, auch die Mitarbeiter in der Systemadministration sind hochzufrieden und erleben eine neue Qualität ihres Arbeitsplatzes. Zudem wurde durch nicht mehr anfallende Lizenzkosten, aber auch durch gesunkene Administrationsaufwände das Ziel einer spürbaren Kostensenkung erreicht. Der nachfolgende Bericht vollzieht die Entscheidungsfindung im Vorfeld der Migration nach, beschreibt Ausgangssituation, Vorgehen und Ergebnis der Umstellung auf Open-Source-Software (OSS) und beleuchtet harte und weiche Faktoren für den Erfolg einer umfassenden OSS-Migrationsstrategie in mittelständischen Unternehmen. Technische, wirtschaftliche und soziale Komponenten werden in Betracht gezogen, und eine Checkliste für die strategische Einführung freier Software wird vorgestellt. Der vor allem an Zusammenhängen interessierte Leser kann die stark technischen Darstellungen des Abschnitts 4 überspringen.

1 *Troubleshooting* steht für das Auffinden und Beseitigen von Fehlern und Störungen.

1. Ausgangssituation, Motivation und Ziele für eine neue IT-Gesamtstrategie

Die NA ist der weltgrößte Kupfer-Recycling-Verwerter und Europas größter Kupferhersteller. Die Konzernstruktur ist vorwärtsintegriert, d. h. von der Kupfererzeugung bis zur Kupferweiterverarbeitung liefert die NA alles aus einer Hand. In den vergangenen drei Jahren ist das Unternehmen und mit ihm seine IT dynamisch gewachsen, und etliche neue Standorte in der Bundesrepublik wurden integriert.

Ständig steigende Systemzahlen, die enorme Breite der eingesetzten Software–Lösungen (Novell, Microsoft, verschiedene Linux-Distributionen wie Mandrake, Red Hat, SuSE) und ein dieses Szenario selbstverständlich begleitender Kostendruck stellten Mitte 2003 die Ausgangssituation für die Evaluierung einer Migration der Datei- und Verzeichnisdienste dar. Während aus wirtschaftlicher Sicht Einsparungen u. a. bei den Lizenzkosten anvisiert wurden, galt das technische Augenmerk der Erhöhung der Interoperabilität[2] vor allem bei der Übernahme fremder IT-Infrastrukturen. Als Ergebnis einer Homogenisierung[3], aber auch als Ergebnis der auszuwählenden Lösung selbst wurde eine Steigerung der Systemverfügbarkeit verlangt. Angesichts des im ersten Ansatz vor allem aus Kostensicht attraktiven Lizenzmodells freier Software lag es nahe, die Lösungen der OSS-Community in die Prüfung der Alternativen einzubeziehen.

2. Entscheidungsvorbereitung

Auf der Wunschliste der IT-Verantwortlichen der NA stand eine möglichst einfache, aber komplette technische Lösung, die bei sinkenden Kosten eine deutliche Erhöhung der integrativen Fähigkeiten auf der Ebene der Infrastruktur und der Anwendung mit sich bringt. Herstellerbindungen galt es auf das Notwendigste zu reduzieren, während gleichzeitig ein breites Dienstleistungs- und Support-Angebot bezogen auf die zu findende Lösung eine wichtige Bedingung für den produktiven Einsatz darstellte.

Die möglichen Optionen wurden einer eingehenden Analyse unterzogen, deren Ergebnis sich im Netzdiagramm (Abbildung 1) widerspiegelt. Während freie Software auf der Ebene der Lizenz- und Wartungskosten deutlich punktete, waren die Personalanforderungen für die Pflege einer Linux-Infrastruktur höher anzunehmen. Hier wurden vor allem Schulungskosten berücksichtigt, aber auch das Gehaltsniveau von Administratoren mit entsprechendem Wissen floss in die Betrachtung ein.

Mit Fokus auf den technischen Bereich wurde eingeordnet, wie nah die möglichen Lösungswege an die Anforderungen des Unternehmens bezogen auf Muss-, Soll- und wünschenswerte Kriterien herankommen. Der Grad der notwendigen Prozessanpassungen drückt aus, wie weit das Unternehmen umgekehrt im Rahmen einer Umstellung auf die Software zugehen muss. Das komplexeste Produkt erhielt hier

2 Interoperabilität beschreibt die Möglichkeit, dass Systeme (Hard- oder Software) unterschiedlicher Hersteller ohne hohen Aufwand zusammenarbeiten können.

3 Homogenisierung meint hier, dass Systeme unterschiedlicher Hersteller vermieden bzw. reduziert werden sollen.

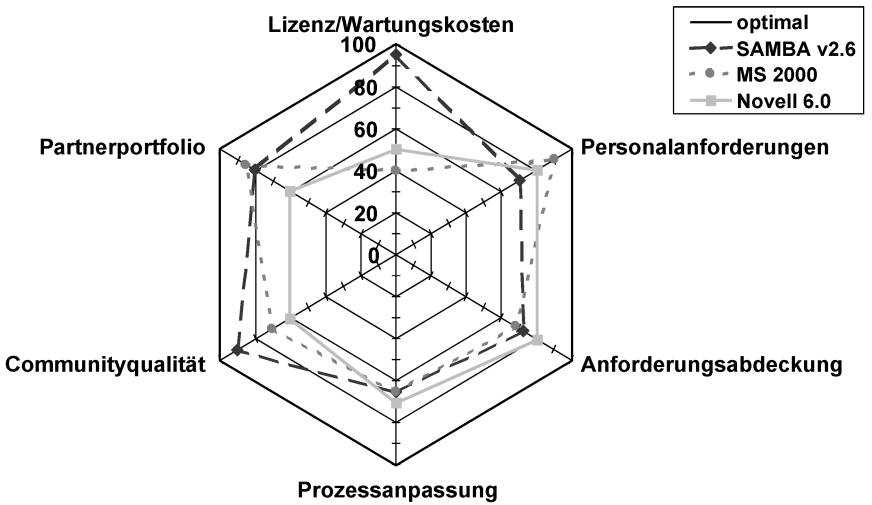


Abbildung 1: Softwareanalyse

die besten Noten, tatsächlich sind die Abstände zwischen den Lösungen aber nach Abgleich mit den tatsächlichen Anforderungen nicht mehr sehr deutlich.

Für ein Unternehmen wie die NA ist es von großer Bedeutung, in welcher Qualität und Breite externe Unterstützung verfügbar ist, sei dies in Form von Dienstleistungspartnern oder der Hilfe anderer Anwender und/oder Entwickler, die in einer Community[4] organisiert sind. Um diese Faktoren zutreffend beurteilen zu können, bedarf es einer langfristigen Beobachtung von Produkten und Anbietern. Als etabliertes Projekt in der Welt freier Software sticht Samba mit einer sehr regen und leistungsfähigen Community heraus. Anzahl und Fähigkeiten der zur Verfügung stehenden Dienstleister mit dem jeweils geforderten Know-how sprachen für eine mit einem Wechsel einhergehende deutliche Verbesserung. Hierbei ist anzumerken, dass die Zahl der Anbieter mit tiefem Samba-Know-how geringer ist als diejenige der auf Microsoft-Produkte spezialisierten Anbieter. Entsprechend wurde also die Qualität und vor allem die Breite des Leistungsportfolios der Partner mit Fokus auf freie Software deutlich höher eingeschätzt.

Das Ergebnis einer weitreichenden strategischen Entscheidung der IT-Abteilung ist natürlich auch auf nicht-technischer Ebene im Unternehmen zu kommunizieren. Während einerseits die Anbieter proprietärer Produkte stark für diese warben, bestätigte andererseits eine befragte Unternehmensberatung die grundsätzliche Tauglichkeit eines Szenarios auf der Basis freier Software für den Unternehmenseinsatz. Auch die

4 Unter Community werden hier auch die Entwicklergemeinschaften im Closed-Source-Bereich verstanden.

zunehmende Bekanntheit der Marke „Linux“ förderte die schließlich im März 2003 auf Vorstandsebene getroffene Entscheidung für die Evaluierung der Einführung von Samba und OpenLDAP als Datei- und Verzeichnisdienst, zunächst als Ersatz der Novell-Infrastruktur in der Konzernzentrale.

3. Evaluierungsphase

Im Juni 2003 begann die NA mit der Umsetzung der geplanten Migration in Form einer Auswahl geeigneter Dienstleister. Gefragt war ein strategischer externer Partner, der die wichtigsten Kriterien und idealerweise auch einige optionale Anforderungen erfüllte. Die Liste der Muss-Kriterien führte hierbei die Ortsnähe an. Immerhin galt es, die eigenen IT-Mitarbeiter so wissensbildend in das Projekt einzubinden, dass diese nach der Umstellung in der Lage sein würden, die neue Infrastruktur im Tagesbetrieb zu warten. Der zukünftige Partner musste zwingend ein sehr breites Portfolio im Bereich OSS vorweisen und durfte auch nicht auf eine bestimmte Distribution festgelegt sein. Insbesondere alle bereits im Einsatz befindlichen Linux-Derivate waren abzudecken. Entsprechende Referenzen wurden abgefragt, und natürlich sollte die externe Unterstützung bei all dem auch kostengünstig und flexibel erfolgen. Wenn dann noch die Fähigkeit zum Blick über den „technischen Tellerrand“ geboten wurde, konnte der strategische Partner als gefunden gelten. Insbesondere gefragt war hierbei Fachwissen rund um Microsoft-basierte Netzwerke und Server-Produkte (wie MS Exchange) und kommerzielle Produkte wie Oracle im Datenbankbereich.

Zwar gelang die Identifizierung des geeigneten Anbieters nicht auf Anhieb, aber im August 2003 war die Suche erfolgreich abgeschlossen. Die auf freie Software spezialisierte inmedias.it GmbH erhielt den Auftrag, eine umfassende Testinstallation vorzunehmen. Die durchgeführten Tests wiesen nach, dass die Portierbarkeit der vorhandenen Struktur auf Samba/OpenLDAP (zunächst unter Laborbedingungen) gegeben war. Bestätigt wurde insbesondere die Machbarkeit der Übernahme komplexer Skripte, Berechtigungen, Nutzdaten und User-/Gruppeninformationen aus der vorhandenen Novell-Infrastruktur mit Hilfe der Standardschnittstellen des *Novell Directory Service* (NDS).

Die IT-Mitarbeiter der NA waren in die Evaluierung von Beginn an eingebunden. Entsprechend gelang der Wissenstransfer, sodass auch die Frage der Administrierbarkeit der zukünftigen Infrastruktur positiv beantwortet werden konnte. Die Mitarbeiter, die bis dahin nur über Linux-Grundkenntnisse verfügten und mit ihrem Fachwissen eher in der Microsoft- und Novell-Welt verortet waren, fanden sich in den neuen Strukturen schnell zurecht. Unterstützt wurden sie hierbei u. a. von webbasierten Verwaltungstools (z. B. LAM), die von der inmedias.it an die Bedürfnisse der NA angepasst wurden.

Mit dem richtigen Partner, nachgewiesener technischer Machbarkeit und der Akzeptanz der zukünftigen Infrastruktur bei den technischen Mitarbeitern konnte im September 2003 die endgültige Entscheidung zur Umsetzung des in Abbildung 2 dargestellten Migrationsfahrplans herbeigeführt werden. Bedingung war hierbei, den Umstieg für die ca. 1 000 Anwender trotz des 24-Stunden-Betriebs möglichst unauf-

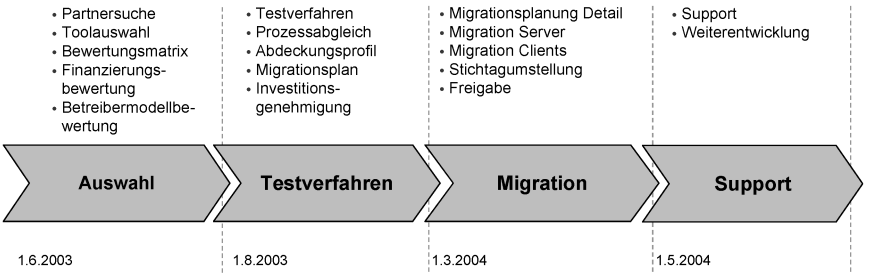


Abbildung 2: Migrationsfahrplan, Ziel: integrierte und kostengünstige File/Print-Lösung

fällig zu gestalten. So wurde als Ziel eine „stille" Migration über die Osterfeiertage 2004 definiert.

4. Vorbereitung

Die Aufgabenstellung bestand vor allem darin, einen konsistenten Übergang von Nutzerinformationen, -rechten und ca. einem Terabyte Nutzdaten zu gewährleisten. Als besondere Herausforderung kam die Übernahme einer äußerst komplexen Struktur von Login-Skripten aus dem NDS hinzu. Die Grundlage der neuen Infrastruktur bildet ein Standard-Server-System mit zwei Prozessoren, auf dem *Red Hat Enterprise Linux Version 3 Advanced Server* (AS) installiert wurde.

Um die Datenübernahmen testen und zum Stichtag „auf Knopfdruck" durchführen zu können, wurden vom externen Partner umfangreiche Skripte entwickelt. Weitere Skripte dienten dem Konsistenznachweis, wobei die transferierten Daten und Rechte mit den ursprünglichen zu vergleichen waren. Samba als Dateidienst wurde unter anderem durch den Einsatz in der Linux-Welt nicht ganz alltäglicher erweiterter Zugriffsrechte, so genannter *Access Control Lists* (ACLs), auf den Bedarf der NA zugeschnitten. Eine webbasierte Administrationsoberfläche wurde geschaffen, wobei der im Red Hat AS enthaltene Webserver durch eigens erstellte Pakete individuell angepasst wurde. Für das als Verzeichnisdienst eingesetzte OpenLDAP mussten Schemata definiert werden, die eine spätere Integration weiterer und derzeit in anderen Verzeichnisdiensten beheimateter Anwendungen vorsehen. Einen Überblick über die einzelnen Schritte und verwendeten Tools bei der Migration verschafft Abbildung 3.

Im Zuge der Migrationsvorbereitung wurden zunächst einzelne, den Standard des Unternehmens repräsentierende Clients in einer ausgedehnten Testphase auf ihr Zusammenspiel mit der neuen Architektur getestet. Das Verhalten der in den typischen Anwendungsfeldern verwendeten Software sowie die mit der Migration verbundenen Arbeitsabläufe wurden ermittelt. Auf Basis dieser Erkenntnisse fand die Umstellung auf das neue System zunächst für „unkritische" Abteilungen statt – u. a. für die IT–Abteilung selbst sowie Abteilungen, die wenig mit Datei-Servern arbeiten.

Während dieser Phase konnten die Clients und die Server-Konfiguration optimal

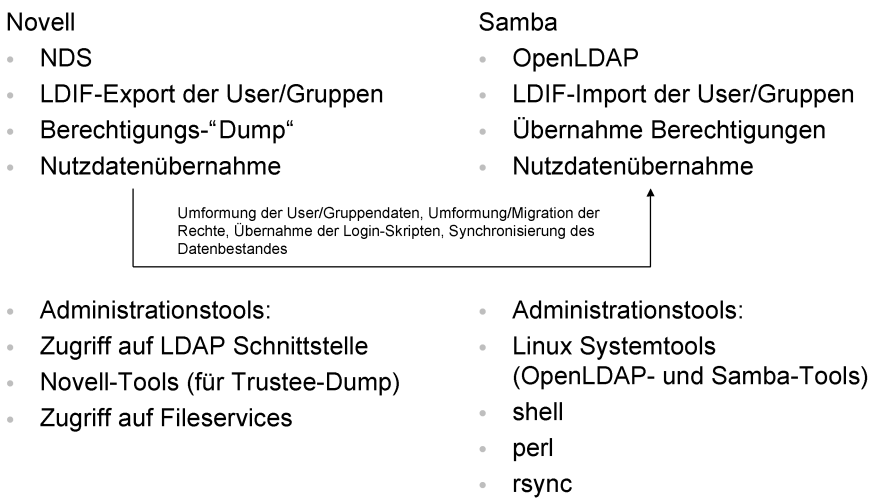


Abbildung 3: Migrationspfade

aufeinander abgestimmt werden, sodass der Abschluss dieser Phase den Weg für die eigentliche Migration freimachte. Die Client-Migrationsstrategie wurde nun in enger Zusammenarbeit beider Seiten erstellt, wobei die Rahmenbedingungen extern vorgegeben wurden. Die Arbeit an den von der NA standardisierten Clients und die Beschaffung und Organisation der nötigen Arbeitskräfte wurden von der NA geplant und durchgeführt.

5. Umstellung

Einige Wochen vor der für Ostern 2004 geplanten „stillen" Migration waren die Vorbereitungen abgeschlossen. So konnte am Vorabend der Datenbestand auf den neuen Server migriert werden, wobei während der Umstellung eine ständige Synchronisation mit dem alten Server-Verbund stattfand. Auf diese Weise war für bereits umgestellte und noch umzustellende Clients stets ein konsistenter Bestand gewährleistet. Bei den notwendigen Umstellungen an den 800 Client-PCs traten kaum Probleme auf, sodass die Migration vorzeitig abgeschlossen werden konnte – die Osterfeiertage wurden entgegen der Planungen nicht gänzlich benötigt. Insgesamt konnte die geforderte Geräuscharmut (lies: minimale Stillstandzeiten, minimale spürbare Auswirkungen für die Nutzer) gewährleistet werden. Die Migration darf als äußerst erfolgreich umgesetzt gelten, womit sich die Aufmerksamkeit auf das Systemverhalten im Alltagsbetrieb richtete.

In der Projektplanung waren für die ersten sechs Wochen unter Last intensive Beobachtung und Nachbesserungen vorgesehen. Tatsächlich wurden nur fünf Tage benötigt, um die „Kinderkrankheiten" der neuen Infrastruktur zu heilen. So führte

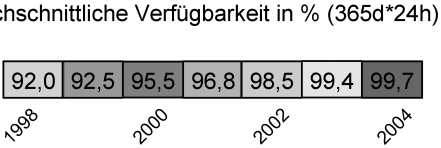


Abbildung 4: Entwicklung der NA-IT in Zahlen I

eine vom Hersteller ungenügend ausgeführte Kernel-Konfiguration zu Performanceproblemen seitens des Red Hat Advanced Servers bei hoher Input-Output-Beanspruchung. Diese Schwierigkeit konnte vom externen Partner durch geeignete Kernelparametrisierung schnell aus der Welt geschafft werden. Ein Kompatibilitätsproblem in der Zusammenarbeit von Microsoft Office 97 und Samba 3.0 führte zu fehlerhaft gesetzten Schreib- und Leserechten. Hier zahlte sich die gute Integration der inmedias.it in die Linux-Community – aus eine ausführliche Fehlerdiagnose versetzte die Samba-Entwicklergemeinschaft in die Lage, das Problem schnell zu bearbeiten. Ein erarbeiteter Workaround und eine nachträgliche Erweiterung des Server-Hauptspeichers waren die bis heute (Stand: September 2004) letzten notwendigen Korrekturen des neuen Systems. Der stabile und fehlerarme Betrieb bewirkt, dass im hausintern geleisteten Anwender-Support keine besonderen Aufwände entstehen – und auch im unmittelbaren zeitlichen Zusammenhang mit der Umstellung nicht entstanden.

6. Technisches Ergebnis

Zwar ist eine IT-Infrastruktur wie so viele andere Dinge nie „fertig“, das Projekt konnte aber im April 2004 abgeschlossen werden und ist in sämtlichen Details als Erfolg zu werten. Im IT-Rechenzentrum der NA versorgt heute ein einzelner Server mit Samba 3.0 und OpenLDAP rund 1 000 Nutzer. Vor der Umstellung wurde die Aufgabe mit vier parallelen Server-Systemen gelöst.

Der Ausfallsicherheit dient ein *Warm Standby Server*, der in regelmäßigen Abständen seine Datenbestände mit denen des Haupt-Servers abgleicht. Die Verfügbarkeit der gesamten, im 365-Tage-/24-Stunden-Betrieb stehenden IT-Infrastruktur ist auf durchschnittliche 99,7% gestiegen (vgl. Abbildung 4). Dies ist vor allem der Stabilität der neu eingeführten zentralen Dienste und einer ab 2002 (es wurden zunächst Datenbankserver auf Linux umgestellt) verfolgten Strategie des verstärkten Einsatzes von Linux zuzuschreiben.

Das zentrale LDAP-Verzeichnis erschließt die Möglichkeit der Integration weiterer Dienste, die derzeit noch separat arbeiten. Umgesetzt wurde diese Integration bereits für den eingesetzten http-Proxy (Squid). Die Einbindung des E-Mail-Servers wird gegenwärtig vorbereitet. Neben der Dienstintegration (mit dem Ziel eines *Single–Sign-On*) ist auch die Standortintegration vorgesehen. Das LDAP-Verzeichnis kann flexibel auf weitere Standorte und Server repliziert werden.

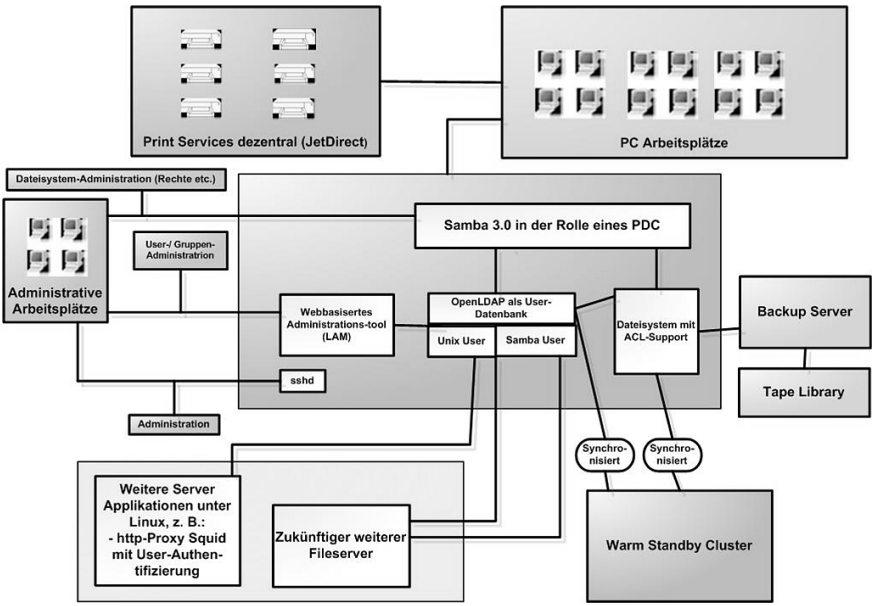


Abbildung 5: File/Print-Dienste nach der Umstellung

Mit der Projektdurchführung ging eine weitere Homogenisierung der gesamten Infrastruktur einher. So hat sich die Zahl der eingesetzten Linux-Distributionen reduziert, und im gesamten Unternehmen finden die Anwender eine einheitliche Windows-Anmeldung (nur noch Windows-Client, nicht mehr Novell/Windows) vor.

Das Backup ist über eine *Tape-Library* (Magnetbandbibliothek) an einem dezidierten Backup-Server realisiert. Für die Druckdienste wurde eine dezentrale Lösung gewählt. Abbildung 5 stellt dar, wie die zentralen Datei- und Verzeichnisdienste sich nach der Umstellung in die IT-Struktur der NA einfügen.

7. Ganzheitliches Ergebnis

Durch eingesparte Lizenzkosten ergibt sich ein sofort messbarer wirtschaftlicher Erfolg der Umstellung. Wie in Abbildung 6 (dunkel unten: Systeme bei der NA, heller darüber: Systeme bei Tochterunternehmen, dunkler oben: Aufwände für externe Dienstleister) dargestellt, konnten die Aufwendungen für Lizenzen trotz stetig steigender Systemzahlen dort stark gesenkt werden, wo freie Software eingeführt wurde. Interessanterweise sind aber auch die Administrationskosten stark gesunken. Die Erwartungshaltung, das zunächst unbekannte neue System würde zu höheren Administrationskosten führen, wurde also erfreulicherweise enttäuscht. Dies ist neben dem gelungenen Wissenstransfer auf die IT-Mitarbeiter der NA vor allem dem problemlo-

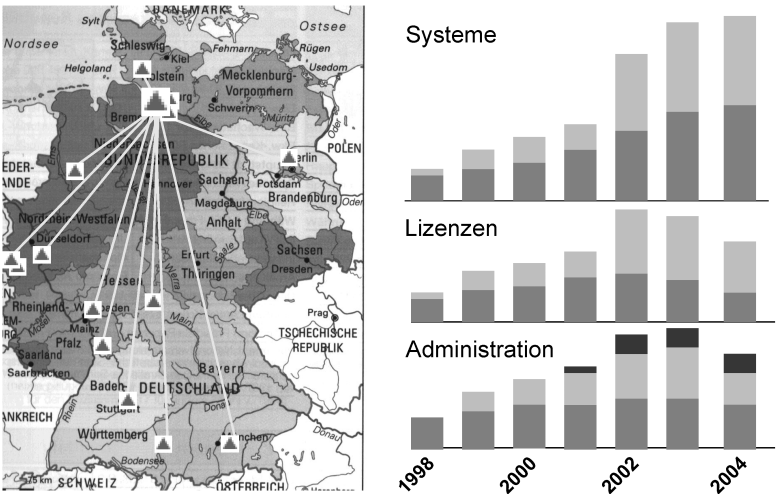


Abbildung 6: Entwicklung der NA-IT in Zahlen II

sen Tagesbetrieb geschuldet. Auch die effektiven Administrationstools tragen ebenfalls zum erfreulichen wirtschaftlichen Erfolg der Umstellung bei – im Gegensatz zur ursprünglichen Erwartungshaltung, die davon ausging, nach der Umstellung bei der Administration weitgehend auf die Kommandozeile angewiesen zu sein. Schließlich wirkt die im technischen Bereich konstatierte hohe Interoperabilität kostensenkend, da sich die technischen Anforderungen einer stark verteilten IT deutlich effektiver erfüllen lassen. Die bei der Migration entstandenen Kosten amortisieren sich im Ergebnis bereits nach knapp neun Monaten.

Zwar bietet die neue Infrastruktur faktisch weniger Funktionalität als die zuvor eingesetzten Novell Directory Services. Das heute eingesetzte Gespann aus Samba und OpenLDAP deckt aber dennoch mehr Funktionen ab, als wirklich benötigt werden. Im Ergebnis ist eine hundertprozentige Erfüllung der tatsächlichen Bedürfnisse einer IT mit ca. 1 000 Nutzern festzustellen. Bezogen auf die zu leistende Integration von verteilten Standorten mit unterschiedlicher technologischer Historie weist die neue Infrastruktur eine Mächtigkeit auf, die von einer heterogenen[5], proprietären Struktur kaum erreicht werden kann. Damit erhöht sich mit der Flexibilität auch die Vorhersagbarkeit von Kosten und Machbarkeit der Integration zukünftiger Unternehmensstandorte und -töchter. Die gewonnene Planungssicherheit ist ein gutes Fundament für eine IT-Strategie, die über die Aufrechterhaltung des Betriebes hinausgeht.

Das Projekt hat aufgezeigt, dass bei der Betrachtung von freier Software nicht ausschließlich technisch eine Liste von Features und wirtschaftlich die einzusparenden Lizenzkosten ein Rolle spielen sollten. Die strategischen Vorteile von OSS sind

5 Heterogen bedeutet hier, dass Produkte/Technologien unterschiedlicher Hersteller in einem System zu finden sind.

dann am besten zu realisieren, wenn deren Entstehungsprozess beobachtet und in die Überlegungen einbezogen wird. Die Existenz einer starken Community, die ein bestimmtes Produkt entwickelt und ein Kontakt zu dieser Entwicklergemeinschaft sind ein wichtiger Erfolgsfaktor beim Einsatz freier Software. Diesen Faktor und technisches Know-how können, wenn das entsprechende Wissen im Unternehmen (noch) nicht vorhanden ist, spezialisierte Dienstleister, wie der im abgeschlossenen Projekt federführende, verfügbar machen.

Die Bildung eines Teams aus den IT-Mitarbeitern der NA und externen Linux-Spezialisten hat einen effektiven Wissenstransfer möglich gemacht. In der Folge konnte ganz auf gesonderte Schulungen für das Fachpersonal verzichtet werden. Bemerkenswert ist hierbei, dass die im Vorfeld der Migration skeptischen IT-Mitarbeiter einen spürbaren Motivationsschub aus den neuen Qualitäten ihres täglichen Arbeitsumfeldes gewonnen haben. Die strategischen Vorteile aus Sicht des Unternehmens decken sich hier in idealer Weise mit den Verbesserungen für die IT-Mitarbeiter.

8. Ausblick und Ableitungen für eine OSS-Gesamtstrategie

Die NA wird die Umstellung ihrer IT auf Linux und freie Software weiter vorantreiben. Zunächst meint dies vor allem die intensive Nutzung der Integrationspotentiale der eingeführten Datei- und Verzeichnisdienste. So wurde bereits ein Unternehmensstandort mit MS-Domänenstruktur (durch eine Vertrauensstellung) mit dem zentralen Samba-Server integriert. Die Ablösung der Dateidienste am Standort und die Aufstellung eines replizierenden OpenLDAP-Servers werden derzeit (Stand: September 2004) umgesetzt. Mit dem durch die Zusammenarbeit mit der inmedias.it gewonnenen Know-how konnten die IT-Mitarbeiter der NA zudem bereits einen kleineren Unternehmensstandort in Eigenregie migrieren.

Doch nicht nur die weitere Homogenisierung der Infrastruktur steht im Fokus der IT-Planung. Auch im Anwendungsbereich, bei der Unternehmensplanung und bezogen auf Desktopbetriebssysteme wird die Entwicklung der einschlägigen OSS–Projekte beobachtet. Hierbei ist gut vorstellbar, dass Softwareentwickler der NA sich an solchen Projekten beteiligen und ihre Beiträge entsprechend dem Prinzip Open Source zur Verfügung stellen.

Freie Software und Linux als Systemplattform können mächtige Tools und Bausteine bei der Gestaltung einer leistungsfähigen Infrastruktur darstellen. Aus der Erfahrung der NA leiten die Autoren fünf Empfehlungen für den Umgang mit OSS ab:

- Beobachten Sie den OSS-Markt,
- Prüfen Sie die Alternativen zu Closed-Source-Software,
- Schätzen Sie das strategische Potential von OSS-Lösungen,
- Suchen Sie sich kompetente Partner,
- Konstruieren Sie Ihr eigenes, umfassendes OSS-Einsatzmodell.

GENOMatch – Datenschutz für die pharmakogenetische Forschung

Broder Schümann und Denis Petrov

GENOMatch ist ein Projekt der Schering AG, das pharmakogenetische Forschung mit einem hohen Datenschutzniveau ermöglicht. Für die Umsetzung und gerade für die Entscheidung, in diesem kritischen Bereich auf Open–Source-Software zu setzen, hat GENOMatch den „Open Source Best Practice Award" vom Fraunhofer IOA, von der Lightwerk GmbH bzw. vom Linux-Verband gewonnen. In dem Artikel wird das Projekt vorgestellt. Dabei wird zunächst der Datenschutz und der daraus resultierende *Workflow* der doppelten Pseudonymisierung genauer beleuchtet. Bei der Beschreibung der Implementierung wird besonders auf die Gründe für den Einsatz von Open–Source-Software und die dabei gewonnenen Erfahrungen eingegangen.

1. Einleitung – Das GENOMatch-Projekt

> „The GENOMatch project is going to provide the IT-infrastructure necessary for pharmacogenetic analyses." (Luttenberger 2003*a*)

Geschrieben wurde dieser Satz Anfang 2003 in einem ersten Entwurf des Datenschutzkonzepts zum GENOMatch-Projekt der Schering AG. GENOMatch stellt den Datenschutz (die *genetic privacy*) der Probanden durch Pseudonymisierungsverfahren sicher.

Seit diesem vollmundigen Versprechen hat sich viel getan: Das Datenschutzkonzept bekam vom Unabhängigen Landeszentrum für Datenschutz Schleswig-Holstein das Auditsiegel[1] verliehen (Luttenberger 2003*b*), das Projekt wurde mit dem vom Fraunhofer IAO, von der Lightwerk GmbH und vom Linux-Verband initiierten „Open Source Best Practice Award" ausgezeichnet (vgl. Ziegler 2004), und die Software ist inzwischen im produktiven Betrieb.

Diesen Weg – von den ersten Ideen über das Datenschutzkonzept bis hin zur Implementierung mit Open-Source-Software, die für so viel Beachtung gesorgt hat – wollen wir in diesem Artikel noch einmal nachvollziehen.

1 Informationen zum Auditsiegel finden sich unter http://www.datenschutzzentrum.de/audit/index.htm.

Dazu geben wir zunächst eine Einführung in die Pharmakogenetik (Abschnitt 2), beschreiben in Abschnitt 3 die Datenschutzziele, umreißen den GENOMatch-*Workflow* (Abschnitt 4) und untersuchen in Abschnitt 5 kurz die Möglichkeiten des technischen Datenschutzes. Im Hauptteil des Artikels (Abschnitt 6) wenden wir uns dann der technischen Umsetzung und unseren Erfahrungen mit Open-Source-Software zu.

2. Überblick – Pharmakogenetische Forschung

> „Pharmacogenetics (PGx), which is now a central focus of pharmaceutical endeavor and on the near horizon of clinical practice, aims to identify genome-wide polymorphisms or mutation that will reliably predict an individual's response to drugs before they are prescribed. Used clinically, PGx information could identify nonresponders and those likely to suffer adverse drug reactions, and thus save them the burden of unsave or ineffective drugs. In addition, PGx information can identify new drug targets and streamline the drug-testing and approval process." (Robertson 2001)

Wie andere Pharmaunternehmen baut auch die Schering AG eine Probensammlung für die pharmakogenetische Forschung auf. Pharmakogenetik zielt darauf, Beziehungen zwischen dem genetischen Profil von Patienten und Wirkungen von Medikamenten zu erforschen.

Um pharmakogenetische Forschung zu ermöglichen, muss eine Korrelation von klinischen Daten mit genetischen Profilen erfolgen. Klinische Daten fallen im Rahmen von klinischen Studien im Alltagsgeschäft der Schering AG an. Die zusätzlich benötigten genetischen Daten werden in Substudien – Erweiterungen bestehender Studien – erhoben. Dieses Vorgehen entkoppelt die beiden Prozesse, sodass der Patient an der klinischen Studie teilnehmen kann, auch wenn er keine Probe für eine pharmakogenetische Untersuchung abgeben möchte. Da in der Bevölkerung genetische Daten als besonders sensibel angesehen werden, hat sich Schering entschlossen, von Anfang an einen Prozess zu entwickeln und zu implementieren, der über die derzeitigen Erfordernisse hinausgeht und zukünftige Gesetzgebungen, soweit möglich, antizipiert. Dieser in Abschnitt 4 kurz vorgestellte Ablauf wurde in „Norbert Luttenberger: Datenschutzkonzept für den 'Sample and Save'-Teil des GENOMatch-Projektes bei der Schering AG" festgeschrieben und vom Unabhängigen Landeszentrum für Datenschutz des Landes Schleswig-Holstein im Rahmen des behördlichen Datenschutzaudits begutachtet. Mit der Implementierung des Konzeptes wurde im April 2003 bei der Tembit Software GmbH begonnen; Ende des Jahres 2004 nahm das System den produktiven Betrieb auf (Abbildung 1).

3. Datenschutz – Erfordernisse und Datenschutzkonzept

Die Erfordernisse des Datenschutzes stellen besondere Anforderungen an das Design eines Systems zur pharmakogenetischen Forschung. Der Europäische Rat sagt in

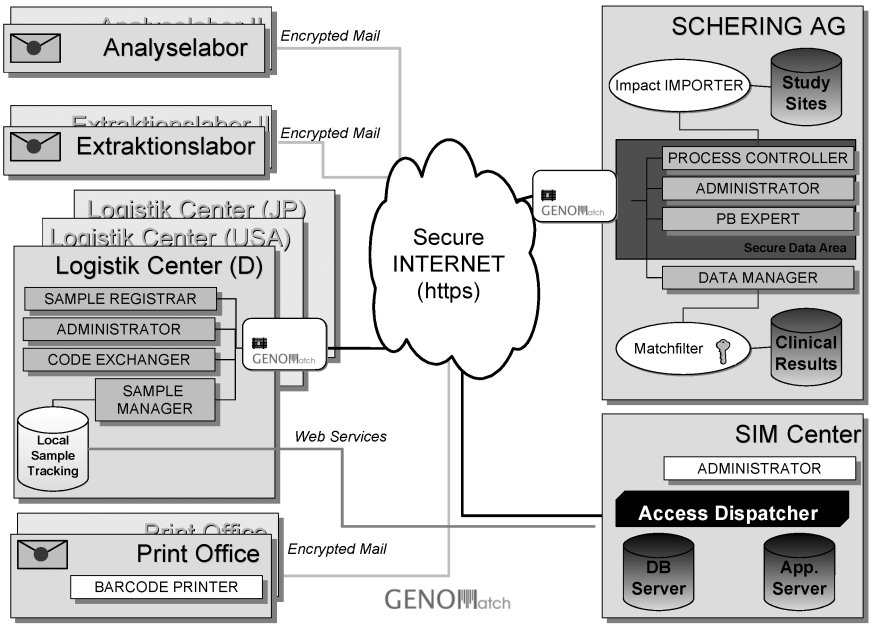


Abbildung 1: GENOMatch-B2B-Interaktionen

Council of Europe (1997) dazu:

> „Die Verarbeitung von Gesundheitsdaten zu Forschungszwecken erfordert zunächst den *Informed Consent* des Patienten und erfolgt am besten vollständig anonymisiert. Können persönliche Daten nicht anonymisiert werden, so sind strikte Datenschutzmaßnahmen nötig."

Eine Anonymisierung würde die Kategorisierung von klinischen Daten vor der Zusammenführung mit den genetischen Datensätzen erfordern und somit den wissenschaftlichen Wert der Proben für die pharmakogenetische Forschung vermindern. Auch die Rücknahme des *Informed Consent* mit der dadurch bedingten Vernichtung von Proben und Datensätzen sowie die Unterrichtung des Patienten über gewonnene Ergebnisse der Studie würden damit unmöglich. Beides widerspricht den Interessen der Forschung und auch den Interessen des teilnehmenden Patienten (Stichwort *Patient Empowerment*). Die *Enquete-Kommission Recht und Ethik in der modernen Medizin* schlägt in Enquete-Kommission (2002) vor,

> „[. . .] ein mehrstufiges Pseudonymisierungsverfahren, möglicherweise mit Verwahrung von Schlüsselbrücken bei Treuhänderinnen bzw. Treuhändern, als Standard für Forschungen mit humangenetischem Material vorzuschreiben."

Hier fordert auch die „Entschließung der 62. Konferenz der Datenschutzbeauftragten des Bundes und der Länder vom 24.–26. Oktober 2001“,

> „[. . .] die Proben und die genetischen Daten vor der Aufnahme in die Sammlung bei Treuhändern zu pseudonymisieren.“ (Datenschutzbeauftragte 2001)

Aus diesen im Datenschutzkonzept von Luttenberger (2003*a*) ausführlich dargelegten Überlegungen heraus setzt GENOMatch auf eine zweifache Pseudonymisierung des Patientenbezeichners – bestehend aus Patientennummer (PN) und Studiennummer. Die zweifache Pseudonymisierung bewirkt, dass zur Auflösung der Pseudonymisierungskette stets mindestens zwei Personen zusammenwirken müssen. Zusätzlich ist die Auflösung des (PN, SN)-Paares zu einer Person wie auch die Kommunikation mit dem Patienten dem behandelnden Arzt vorbehalten.

4. GENOMatch-*Workflow* – Doppelte Pseudonymisierung

Auch wenn es auf den ersten Blick nicht den Anschein hat, so ist der eigentliche GENOMatch-*Workflow* relativ leicht verständlich.

An dem GENOMatch-Prozess nehmen folgende Institutionen teil:

- Schering AG (SAG)
- Central Sample Repository (CSR): Zentrales Sammellabor, zuständig für Lagerung und Logistik der Proben
- Trial Site: Klinik/Arztpraxis, in der Patienten an einer Studie teilnehmen
- Secure Identity Management Center (SIM-Center): Datentreuhänder. Ein Blackbox-Server-System, extern betrieben bei einer Anstalt des öffentlichen Rechts
- Externe Dienstleister: Analyse- und Extraktionslabore, Barcode-Druckereien usw.

Der grundlegende Ablauf funktioniert so, dass die Trial Site die Proben mit (PN, SN) beschriftet an das CSR sendet. Dort findet die doppelte Pseudonymisierung statt: Für jede Probe wird dort die (PN, SN)-Bezeichnung entfernt und durch das erste Pseudonym (in Form eines Barcodes, BC1) ersetzt. Eine andere Person entfernt diesen BC1 und ersetzt ihn durch das endgültige Pseudonym BC2. Die Verbindung der jeweiligen *Identifier* wird im SIM-Center gespeichert. Erst danach können genetische Daten gewonnen werden, die als einzigen *Identifier* den BC2 der jeweiligen Probe tragen. Zur biostatistischen Auswertung werden in einem speziell abgeschirmten Bereich bei SAG (der sog. *Secure Data Area*, SDA) die genetischen mit den klinischen Daten zusammengeführt. Dazu werden die (PN, SN)-indizierten klinischen Daten an der Grenze zur SDA durch einen sog. *Matchfilter* ebenfalls in zwei Schritten pseudonymisiert, sodass sie schließlich keine personenbezogenen Daten, keine PN oder SN, sondern nur noch das BC2 Pseudonym tragen (Abbildung 2).

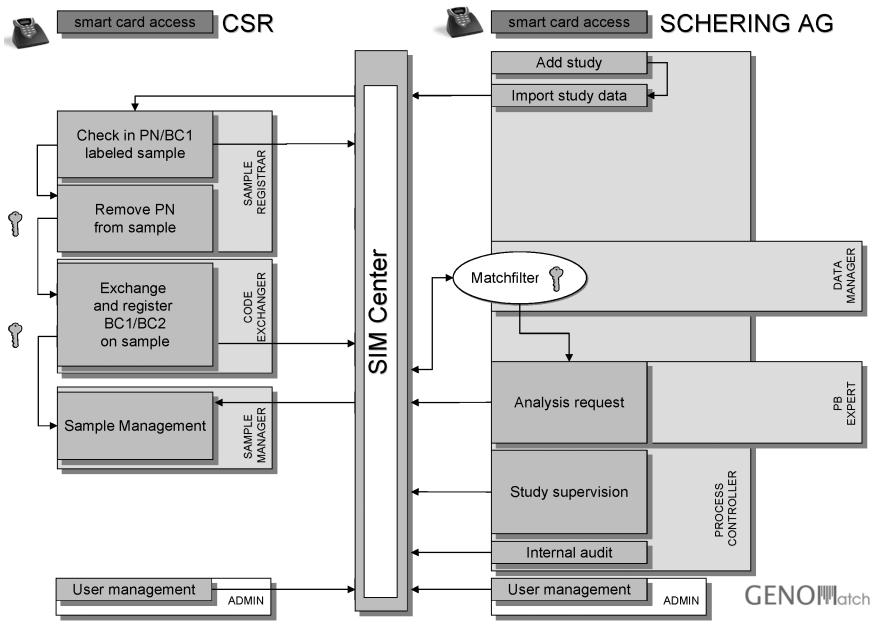


Abbildung 2: GENOMatch-Abläufe

In dieser Beschreibung ausgeblendet wurden viele Details, Sonderfälle, logistische Prozesse sowie Prozeduren zum Feedback von Ergebnissen an den Patienten oder dem Rückzug seines *Informed Consent*. Der GENOMatch-*Workflow* stellt sicher, dass stets folgende Prinzipien gelten:

- Die Identität des teilnehmenden Patienten ist ausschließlich dem Arzt bzw. der Klinik bekannt (Zuordnung PN, SN nach Person).
- Keiner einzelnen Person ist die Auflösung des Pseudonyms BC2 zum (PN, SN)-*Identifier* möglich. Diese Zuordnung ist nur dem SIM-Center bekannt.
- Genetische Daten werden sowohl bei Schering als auch bei externen Dienstleistern nur mit BC2 pseudonymisiert gespeichert und verarbeitet und enthalten keine personenbezogenen Daten.
- Die Zusammenführung der klinischen und genetischen Daten erfolgt in einem abgeschirmten Bereich, nach Löschung aller personenbezogenen Daten und ausschließlich unter dem BC2-Pseudonym.

5. IT-Unterstützung – Technischer Datenschutz

Inwieweit kann und soll ein technisches System diesen *Workflow* unterstützen bzw. durchsetzen? Prof. A. Roßnagel (2004) stellt fest:

> „Außerdem ist technischer Datenschutz viel effektiver als rein rechtlicher Datenschutz. Was technisch verhindert wird oder unterbunden werden kann, muss nicht mehr verboten werden. Gegen Verhaltensregeln kann verstoßen werden, gegen technische Begrenzungen eines Techniksystems nicht."

Ziel des technischen Datenschutzes in GENOMatch ist es also, den Pseudonymisierungsprozess mit technischen Mitteln durchzusetzen, d. h. die beteiligten Personen zu zwingen, von diesem Prozess nicht abzuweichen. Systeme wie GENOMatch erfordern jedoch die Bearbeitung greifbarer Gegenstände durch Menschen (hier z. B. das Entfernen und Anbringen von Etiketten). Solche Systeme müssen darauf vertrauen, dass der Zustand des Gegenstandes dem erwarteten und vorgeschriebenen Zustand entspricht (z. B. „der (PN, SN)-Aufkleber wurde entfernt"). Kontrollieren kann das IT-System diese Annahme nicht. Technische Mittel und verbindliche Verfahrensanweisungen für die beteiligten Personen müssen sich also so ergänzen, dass insgesamt der im Datenschutzkonzept vorgeschriebene *Workflow* sichergestellt wird.

Der technische Datenschutz kann den Prozess allerdings in vielen Bereichen unterstützen. Insbesondere eine strikte rollenbasierte Zugangskontrolle zu Funktionen des Systems und somit auch zu den jeweils verfügbaren Daten gewährleistet eine zuverlässige Pseudonymisierung. Zum Beispiel existiert eine Probe aus Sicht der Mitarbeiter bei Schering überhaupt erst dann, wenn sie vollständig pseudonymisiert wurde. Die Pseudonymisierung dürfen aber nur Mitarbeiter des CSR vornehmen. Somit können nur für pseudonymisierte Proben genetische Daten generiert werden.

6. Open Source – Entscheidung und Erfahrungen

Die Entscheidung, bei der Implementierung auf Open-Source-Produkte zu setzen, stand nicht von vornherein fest. Sie wurde nicht dogmatisch gefällt, vielmehr wurde für jede einzelne Design-Komponente evaluiert, welche Software am besten zur Realisierung der gewünschten Eigenschaften geeignet ist. Dass letztendlich im *GENOMatch Application Layer* fast ausschließlich freie Software zum Einsatz kommt, spricht für die Qualität von Open-Source-Software.

6.1. Architektur

Der grundsätzliche *Workflow* des doppelten Pseudonymisierungsprozesses mit einem Datentreuhänder diktiert große Teile der Systemarchitektur. Der Datentreuhänder – das SIM-Center – verwaltet alle Informationen, stellt die Pseudonymisierung sicher und macht beteiligten Personen stets nur die für sie bestimmten Daten zugänglich. Auf diesen Rechner greifen Nutzer aus mehreren Institutionen über das Internet zu,

Umgebung	Betriebssystem	Server-Software	Programmiersprache
Microsoft	Windows	Internet Information Server (IIS)	.NET (VB, C, C++, C#)
Open Source	Linux	Apache/ mod_ssl/ J2EE-Applikationsserver	Java

Tabelle 1: Betrachtete Standardumgebungen

sodass sowohl eine Standardisierung der Schnittstellen als auch eine Verschlüsselung der Kommunikation erforderlich ist.

Den Nutzern bietet der Server eine Weboberfläche an. Dies erlaubt sowohl die Nutzung von normalen Arbeitsplatz-Rechnern aus als auch die Möglichkeit, ohne großen Aufwand an Punkten, wo kritische Daten anfallen, *Thin Clients* einsetzen zu können. Die Authentifizierung mit *Smartcards* stand weit oben auf der Wunschliste und sollte evaluiert werden, galt zunächst jedoch nicht als zwingende Voraussetzung.

Dienste, die von Programmen (wie dem *Matchfilter* oder der lokalen Lagerverwaltungs-Software des CSR) in Anspruch genommen werden sollen, können mit vielen Techniken realisiert werden: Dateitransfers, *Remote Method Invocation* (RMI), *Common Object Request Broker Architecture* (CORBA) oder *Web Services* kamen hier in Frage. Für *Web Services* sprechen viele Gründe: Standardisierung, Zukunftssicherheit und Plattformneutralität; vor allem aber die Bündelung der kompletten Kommunikation auf ein verbreitetes und sicheres Protokoll (*Hypertext Transfer Protocol over Secure Socket Layer*, HTTPS), das keine Spezialkonfiguration in Firewalls benötigt.

Diese Architektur und diese Schnittstellen standen schon fest, bevor das erste technische Kick-off-Meeting stattfinden sollte. Über Betriebssysteme, Server-Software, Browser, Programmiersprachen und Tools war bis dahin noch kein Wort verloren worden, weshalb diese Besprechung von allen Seiten mit Spannung erwartet wurde (Abbildung 3).

6.2. Server-Seite – SIM-Center

Die wohl grundlegendste Entscheidung betraf die Programmiersprache. Diese bestimmt dann auch die Server-Software und damit praktisch auch das Betriebssystem für das SIM-Center. Um die im vorherigen Abschnitt beschriebene Architektur zu realisieren, gibt es eigentlich nur zwei Standardumgebungen:

Eine Reihe von Gründen sprach für die Open-Source-Lösung. Zu der höheren Sicherheit einer minimalisierten Apache-Installation kam vor allem die Möglichkeit, Anpassungen im Bereich der Verschlüsselung und der Authentifizierungsmechanismen vorzunehmen. Beides sind Faktoren, die durch die Verfügbarkeit des Quellcodes bedingt oder zumindest gefördert werden.

Wie in jedem Projekt entscheiden natürlich auch Entwicklungszeit und -kosten für das eine oder andere System. Mit den genannten Open-Source-Produkten waren um-

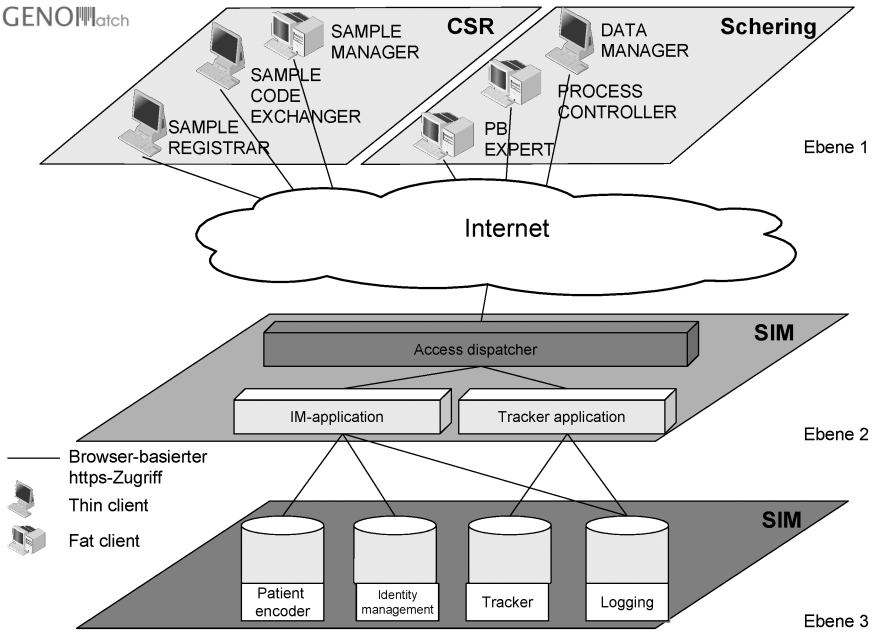


Abbildung 3: GENOMatch-Architektur

fangreiche Erfahrungen vorhanden. Aber auch die Möglichkeit, selbst Fehler zu finden und zu bereinigen oder selbst die nötigen Erweiterungen hinzuzufügen, beschleunigen den Entwicklungsprozess, da die Wartezeiten auf *patches* oder Fehlerdiagnosen des jeweiligen Herstellers entfallen.

Diese Grundsatzentscheidung für Linux/Apache/Java sorgte dann dafür, dass der Rest der Software-Komponenten in diesem Systemumfeld schnell gefunden war. Für den Server boten sich viele der ausgezeichneten Tools der Apache Foundation an:

- Java-Applikationsserver: Tomcat (Apache Jakarta Project)
- *Web Services*: Apache Axis
- *Logging*: log4j
- Weboberfläche: Struts
- *Build/Deployment*: Ant

Ergänzt werden diese Tools durch JavaMail von Sun Microsystems (für den Mailversand) und BouncyCastle Security Provider von BouncyCastle (für die Mailverschlüsselung). Auch Software zur Entwicklung wie Eclipse und CVS sind hervorragende und vielfach erprobte Tools.

Einzig bei der benötigten SQL-Datenbank fiel die Wahl dann wieder auf ein kommerzielles Produkt: Oracle. Oracle ist bei Schering der Standard für Datenbankanwendungen. Weitere Gründe waren vor allem die Stabilität, die Kompatibilität bei Migrationen zwischen Versionen und der Support. Diese waren vor allem deshalb entscheidend, weil die Pseudonymketten für einen Zeitraum von 20 bis 25 Jahren verfügbar bleiben müssen.

Die Wahl der Komponenten hat sich als ausgezeichnet erwiesen und hat es ermöglicht, schnell ein stabiles System mit allen gewünschten Merkmalen zu realisieren.

6.3. Thin Clients

Es gibt mehrere Stellen, an denen im GENOMatch-Prozess sensible Daten anfallen. Dies ist insbesondere bei der eigentlichen Pseudonymisierung im CSR der Fall. Die hier eingegebenen Bezeichnerpaare dürfen niemals außerhalb des SIM-Centers gespeichert werden.

Die Dateneingabe findet deshalb auf *Thin Clients* statt. Diese Geräte verhindern sowohl ein Speichern der Daten als auch jegliche Veränderung am System (wie beispielsweise *Keylogger* oder Screenshot-Programme). Auch Netzwerkverbindungen mit anderen Rechnern als dem SIM-Center müssen unterbunden werden.

Technisch umgesetzt wurden diese Anforderungen mit einem minimierten Linux–System. Dieses System bootet von einer CF-Karte, die ausschließlich lesbar eingebunden wird. Alle Dateien, die veränderbar sein müssen, werden auf RAM-Disks im Arbeitsspeicher gehalten und sind somit durch Ausschalten des Geräts zuverlässig zu löschen. Eine Grundkonfiguration dieser RAM-Disks wird beim Systemstart automatisch aus Konfigurationsimages erzeugt. Ein Paketfilter sorgt dafür, dass die Systeme ausschließlich über die vorgesehenen Protokolle (hier: HTTPS) mit bestimmten Rechnern (hier: SIM-Center) kommunizieren können.

Die Geräte selbst erfordern keine Nutzerauthentifizierung, sondern starten einen X-Server und einen Browser (Mozilla Firefox). Andere Software ist nicht installiert bzw. nicht startbar. Der Nutzer weist sich über das Netzwerk gegenüber dem SIM–Center mit seiner *Smartcard* aus, die mittels eines Browser-*Plugins* das *Secure-Socket–Layer*- (SSL) Client-Zertifikat für die HTTPS-Verbindung bereitstellt.

Der Flexibilität und Einfachheit von Linux ist es zu verdanken, dass ein erstes funktionierendes System innerhalb kürzester Zeit erstellt werden konnte. Die Installation, inklusive aller Anpassungen für den oben beschriebenen *read-only*-Betrieb, nahmen so weniger als zwei Tage in Anspruch. Das ist weniger Zeit, als für die vorher unternommenen Versuche, eine Browser/*Smartcard*-Kombination mit geschlossenen Systemen zu realisieren benötigt wurde. Eine spezialisierte Lösung auf Basis von Open Source zu erstellen, bietet mehr Freiheit, Kontrolle und Eingriffsmöglichkeiten, als der Versuch, eine proprietäre Lösung anzupassen.

6.4. Smartcards

Ein Grundsatz bei GENOMatch ist, dass jegliche Netzwerkkommunikation verschlüsselt erfolgt. Dies betrifft unter anderem folgende Kommunikationskanäle:

- Nutzung des SIM-Centers über die Weboberfläche. Hier kommt HTTPS (SSL–verschlüsseltes *Hypertext Transfer Protocol*) zum Einsatz.
- Zugriffe von externen Programmen oder GENOMatch-Komponenten auf das SIM-Center. Diese sind als *Web Services* realisiert; als Protokoll wird ebenfalls HTTPS genutzt.

Die Verschlüsselung aller Zugriffe auf das SIM-Center verhindert das Abhören der übertragenen Daten durch Dritte. Es bleibt jedoch für einen Angreifer möglich, den Netzverkehr umzuleiten und sich dem Nutzer gegenüber als SIM-Center auszugeben. Dem wird durch die Verwendung von digitalen Zertifikaten begegnet. Das SIM-Center weist sich mit einem solchen Zertifikat aus, wie es auch z. B. beim Homebanking über eine Weboberfläche Stand der Technik ist.

Damit kann der Nutzer sicher sein, mit dem SIM-Center zu kommunizieren. Die Nutzerauthentifizierung wird üblicherweise jedoch immer noch mittels Nutzerkennung und Passwort durchgeführt. Die beiderseitige Identitätsprüfung mittels Zertifikaten verlangt auch vom Nutzer das Vorweisen eines digitalen Ausweises. Dieser wird bei GENOMatch auf einer *Smartcard* gespeichert und ist mit einer PIN geschützt.

Dadurch wird die Sicherheit in mehreren Bereichen verbessert:

- Das Zertifikat verlässt niemals die *Smartcard*, kann also nicht kopiert werden.
- Zugang zum System erfordert nicht nur das Wissen um ein geheimes Passwort / PIN, sondern auch den Besitz einer *Smartcard*.
- Die Eingabe der PIN erfolgt direkt am Kartenleser und ist somit auch an *Fat Clients* vor Mitschneiden geschützt.

Leider werden digitale Zertifikate auf einer *Smartcard* noch nicht in großem Umfang eingesetzt, obwohl das Gesetz zur *Digitalen Signatur* dafür schon länger Rahmenbedingungen vorgibt. Dies bewirkt, dass viele Standardprodukte diese Technik nicht unterstützen und sich ein Markt für proprietäre Nischensysteme gebildet hat.

Die Nutzung von *Thin Clients* stand von Beginn an fest, und es wurde früh überlegt, hier evtl. auf Linux zu setzen. Auf der CeBIT 2003 wurde deshalb bei mehreren Herstellern von Kartenlesern nachgefragt, ob diese ein Browser-*Plugin* und einen Treiber auch für Linux bereitstellen. Viele Hersteller bieten ihre Lösung als *Login*-Mechanismus für Windows oder als komplett proprietäre Lösung an, unterstützen jedoch keine SSL-Client-Zertifikate für Browser. Diejenigen, die diese Möglichkeit vorsehen, liefern jedoch meistens nur *Plugins* für Windows Internet Explorer, auf Nachfragen wurde als Grund oft genannt, dass für die Unterstützung von Netscape/Mozilla und/oder Linux/Unix kein Markt vorhanden sei und es sich somit nicht lohne. Die einzige Firma, die in ihren Geräten Netscape/Mozilla auf mehreren Plattformen unterstützt, ist die Firma Kobil. Hier gibt es gegen Aufpreis Kartenleser mit der benötigten Software sowohl für Windows als auch für Linux und Solaris.

Diese Komponente ist außer der Oracle Datenbank die einzige proprietäre Software, die in GENOMatch zum Einsatz kommt. Zugleich ist dies ein Punkt, an dem

ein benötigtes *Feature* nicht realisiert werden konnte. Nachdem zunächst der Hersteller zugesagt hatte, dieses Merkmal einzubauen, hat man sich nun doch dagegen entschieden. Es bleiben somit nur die Möglichkeiten, den Hersteller dafür in einem gesonderten Auftrag zu bezahlen oder in Verhandlungen die Herausgabe des Quellcodes zu erreichen. Hier sind bisher noch keine Fortschritte erzielt worden. Hätte es sich um offene Software gehandelt, so wäre die Erweiterung mit einer Entwicklungszeit von maximal einer Woche durchführbar gewesen und hätte danach der gesamten Community zur Verfügung gestanden.

7. Fazit

GENOMatch ist nicht nur unter den Aspekten des Datenschutzes und der damit verbundenen Prozeduren ein interessantes Projekt. Auch technisch mussten viele Hürden genommen werden, bei denen uns die Flexibilität, ein Blick in den Quellcode und die hervorragende Dokumentation im Internet – seien es *HOW-TO*s, Erfahrungsberichte oder Foren und Mailinglisten – oftmals weitergeholfen haben.

Dass wir fast ausschließlich Open-Source-Produkte einsetzen, und dass dies nicht aus politischen, sondern allein aus technischen Gründen geschieht, zeigt die Reife und Qualität Freier Software.

Wir glauben, mit diesen Mitteln ein qualitativ hervorragendes Produkt abgeliefert zu haben, das mit anderer Software so vielleicht nicht realisierbar gewesen wäre.

Literatur

Council of Europe (1997), 'RECOMMENDATION No. R (97) 5 OF THE COMMITTEE OF MINISTERS TO MEMBER STATES ON THE PROTECTION OF MEDICAL DATA', COUNCIL OF EUROPE COMMITTEE OF MINISTERS, http://cm.coe.int/ta/rec/1997/97r5.html [15. Jan 2005]. Adopted by the Committee of Ministers on 13 February 1997 at the 584th meeting of the Minsters' Deputies.

Datenschutzbeauftragte (2001), 'Anlage zu „Umgang mit genetischen Untersuchungen"', Datenschutz Berlin, http://www.datenschutz-berlin.de/doc/de/konf/62/anlage.htm [15. Jan 2005]. Entschließung der 62. Konferenz der Datenschutzbeauftragten des Bundes und der Länder vom 24.-26. Oktober.

Enquete-Kommission (2002), 'Schlussbericht der Enquete-Kommission „Recht und Ethik in der modernen Medizin"', Deutscher Bundestag, http://dip.bundestag.de/btd/14/090/1409020.pdf [15. Jan 2005]. Bundestagsdrucksache 14/9020.

Luttenberger, N. (2003*a*), 'Data Protection Concept for the "Sample and Save" Part of the GENOMatch Project at Schering AG', (erhältlich auf Anfrage).

Luttenberger, N. (2003*b*), 'Kurzgutachten zum Datenschutzaudit: Konzept einer Datenverarbeitungsinfrastruktur der Fa. Schering AG für die sichere pseudonyme Einlagerung und Verwahrung von für genetische Analysen genutzten Blut- und Gewebeproben', Unabhängiges Landeszentrum für Datenschutz Schleswig-Holstein,

http://www.datenschutzzentrum.de/audit/kurzgutachten/a0303/index.htm [15. Jan 2005]. Entwickelt von der AG Kommunikationssysteme am Institut für Informatik und Praktische Mathematik, Prof. Norbert Luttenberger, Christian-Albrechts-Universität zu Kiel.

Robertson, J. A. (2001), Consent and privacy in pharmacogenetic testing, *in* 'nature genetics', Vol. 28, Nature Publishing Group, S. 207–209. zu beziehen unter http://www.nature.com/cgi-bin/doifinder.pl?URL=/doifinder/10.1038/90032 [15. Jan 2005].

Roßnagel, A. (2004), 'Datenschutzrecht in Deutschland', Friedrich-Ebert-Stiftung Korea, http://www.fes.or.kr/Publications/pub/Rossnagel_PSPD-2004.pdf [30. Jan 2005]. Vortrag für den Korean-German Joint Workshop on Privacy Protection der People's Solidarity for Participatory Democracy und der Friedrich-Ebert-Stiftung am 1. November 2004 in Seoul.

Ziegler, P.-M. (2004), 'Sieger des ersten "Open Source Best Practice Award" präsentiert', heise online, http://www.heise.de/newsticker/meldung/51885 [15. Jan 2005].

Weiterführende Informationen

Informationen zur Software der Apache Software Foundation finden sich unter (Tomcat, Apache Axis, log4j, Struts, Apache Ant sind Warenzeichen der *The Apache Software Foundation, USA.*):

- Tomcat: http://jakarta.apache.org
- Apache Axis: http://ws.apache.org/axis
- log4j: http://logging.apache.org/log4j
- Struts: http://struts.apache.org
- Apache Ant: http://ant.apache.org

Detaillierte Informationen zur JavaMail API finden sich auf den Webseiten von Sun Microsystems (Java, J2EE und JavaMail API sind Warenzeichen der *Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054, USA.*):

- Java: http://java.sun.com/
- J2EE: http://java.sun.com/j2ee/index.jsp
- JavaMail API: http://java.sun.com/products/javamail

Weitere Informationen zum BouncyCastle Security Provider sind auf den Internetseiten des Herstellers zu finden (*BouncyCastle Security Provider – Copyright (c) 2000 - 2004 The Legion Of The Bouncy Castle*): http://www.bouncycastle.org

Kapitel 2

Technik

"The thing he realized about the windows was this: because they had been converted into openable windows after they had first been designed to be impregnable, they were, in fact, much less secure than if they had been designed as openable windows in the first place."

– Douglas Adams: Mostly Harmless

Open Code Worlds

WOLFRAM RIEDEL

1. Einleitung zum Kapitel „Technik“

Unser Buch vereint Betrachtungsweisen aus sehr unterschiedlichen Disziplinen und beinhaltet auch Thematiken, die mit Technik nur noch am Rande zu tun haben. Und doch lohnt es sich, zunächst noch einmal zu den technischen Wurzeln der neuzeitlichen Open-Source-Bewegung zurückzukehren, bevor man den Versuch wagt, die dort bereits funktionierenden Modelle auch auf andere Bereiche zu übertragen. Das Kapitel „Technik“ bietet somit nicht nur eine interessante Lektüre für technisch interessierte Leser, sondern auch eine bereichernde Grundlage für die weiteren fachwissenschaftlichen Vertiefungen.

Innerhalb des letzten Jahres ist viel passiert, und wie immer wird das vergangene Jahr gerne als das beste, aktivste und erfolgreichste Jahr der Open-Source-Geschichte bejubelt (Lindner 2005). Und natürlich ist es nicht möglich, alle aktuellen technischen Entwicklungen in diesem Kapitel abzudecken, jedoch hoffe ich, dass es gelungen ist, die interessantesten Themen herauszupicken. Nach einem Blick auf den Open-Source-Desktop und seine Benutzbarkeit widmen wir uns der Evolution freier Software und der Entwicklung von Linux auf eingebetteten Systemen.

2. Open Source auf dem Desktop

Nach dem stetigen Erfolg von Linux im Server-Segment ist inzwischen immer öfter von Umsteigern bei Desktop-Rechnern zu hören: Die Migrationsberichte häufen sich, bei denen auch die Arbeitsplätze mit Linux „gefahren“ werden. Maßgebliche Faktoren für diese Entwicklung sind die beiden beliebten Desktop-Umgebungen GNOME und KDE, die in den rasch vorangetriebenen Versionen des Jahres 2004 schon sehr weit gereift sind.[1] Und auch OpenOffice, die wohl beliebteste freie Office-Suite, hat mit Sicherheit keinen geringen Anteil am Erfolg von Open Source auf dem Desktop.

1 KDE 3.2 erschien im Februar, KDE 3.3 im August. Vor allem seine PIM-Applikationen wurden stark weiterentwickelt und arbeiten untereinander besser zusammen. GNOME 2.6 ist seit April, GNOME 2.8 seit September 2004 verfügbar. Sowohl das *Panel* als auch der Dateimanager Nautilus sind noch übersichtlicher geworden. Ein weiteres Highlight ist die Groupware-Applikation Evolution 2.0.

2.1. Rückgrat des Desktops

Eine sehr verbreitete und zentrale Komponente auf freien, unixoiden Desktop-Systemen ist der X-Server. Ihm wird im Vergleich zu den darauf aufbauenden Oberflächen KDE und GNOME oft weniger Aufmerksamkeit geschenkt.

Für Wirbel in der Szene sorgte 2004 eine Lizenzänderung der freien Implementation XFree86, deren Auswirkungen und GPL-Kompatibilität unklar und umstritten bleiben. Zahlreiche Distributionen verwenden XFree86 seitdem nicht mehr. Besonders bemerkenswert ist der zeitnahe Umbau der X.Org-Stiftung[2], die nun hauptsächlich von Entwicklern der Community geleitet wird und die die letzte XFree86-Version vor der Lizenzänderung „geforkt" hat und jetzt offen weiterentwickelt.

Als Plattform im Web dient freedesktop.org, die sich als ein zentraler Anlaufpunkt für die Community herauskristallisiert hat, wenn es um Standards, *frameworks* und Backends für den X-Desktop geht. Die Zugewandheit zur Community sorgte anscheinend für einen Motivationsschub: Die neuen Features der ersten Version nach der Neuaufstellung bescherten dem Projekt den Linux New Media Award[3] in der Kategorie „Best Free Project for Hardware Support".

Im Sommer 2004 erschien erstmals FreeNX, eine auf einem Shellskript basierende freie Implementation des NX-Servers von NoMachine[4]. Im X-Protokoll einer quasi nächsten Generation wurde, vereinfacht gesagt, der Kommunikations-Overhead deutlich reduziert und eine Kompression eingebaut.[5]

> „Selbst bei geringer Bandbreite wie mit Modems oder per ISDN erreichen NX und FreeNX eine gute Performanz – und das quer über mehrere Betriebssystem-Plattformen hinweg." (Kurt Pfeifle, zitiert bei Ihlenfeld 2004*a*)

Zügig wurde die Software in Knoppix integriert und steht seit der KDE-Konferenz aKademy 2004 auch auf „temporären" Linuxrechnern ohne großen Installationsaufwand zur Verfügung.

2.2. Fenster-Notausstieg – bitte Scheibe einschlagen

Nach einer möglichen Entscheidung, die bunten Kirchenfenster durch geschäftige Frackträger zu ersetzen, steht der Migrant zum Glück nicht allein auf weiter Flur: Es gibt zahlreiche Hilfsmittel, die einen Umstieg von Windows auf Linux bzw. die Einarbeitung in Linux erleichtern können. Spezielle Linux-Distributionen bereiten gerade weniger erfahrenen Computernutzern weniger Probleme beim Umstieg und der

2 X.Org http://www.x.org/

3 Linux New Media Award 2004 http://www.linuxnewmedia.de/Award_2004

4 NoMachine http://www.nomachine.com

5 Die Netzwerk-Fähigkeit erlaubt das Arbeiten auf entfernten Rechnern. Bei NX und FreeNX handelt es sich nicht um einen offiziellen Standard der X.Org-Stiftung. Es ist aber anzunehmen, dass die zukünftige Entwicklung des X-Protokolls von X.Org in diese oder eine ähnliche Richtung gehen wird.

sich anschließenden Benutzung. Neben den schon immer als recht benutzerfreundlich geltenden Distributionen von Novell [6] und Mandrake[7] wirbt besonders Xandros[8] mit einem intuitiven Desktop. Einige Umsteiger, die sehr an die Microsoft'schen Oberflächen gewöhnt sind, tun sich schwer, sobald ein Desktop eine andere Optik anbietet. Linspire[9] und Freedows[10] sind zwei Distributionen, die diese Nutzergruppe ins Auge gefasst haben und sich explizit als Windows-Ersatz platzieren. Ihre Produkte ähneln einem Windows-Desktop und mindern so Berührungsängste beim Umstieg. Zusätzlich bieten sie wie Xandros teilweise Kompatibilität zu gängigen Windows–Programmen. Weiterhin gibt es im Internet eine Fülle von Anleitungen und Tutorials für den geschäftlichen wie auch den privaten Umsteiger, z. B. die IOSN-Anleitung[11] oder das Linux Client Migration Cookbook[12] von IBM. Sogar Schulungs-Videos[13] sind frei verfügbar.

2.3. Feuerfuchs und Donnervogel

Mozilla, das vom Markt verdrängte Stiefkind des einstigen Browserpioniers Netscape, hat sich zur sprichwörtlichen Eier legenden Wollmilchsau gemausert. Und gleichzeitig ist das Projekt ein leuchtendes Beispiel für die Möglichkeiten und die Flexibilität von OSS: Der nur schwer wartbare Netscape-Code wurde nach der Veröffentlichung 1998 von der Community einem grundlegenden *Re-Design* unterzogen, bei dem sich der Änderungsaufwand auf ein Fünftel und der Koordinationsaufwand auf ein Drittel reduzieren ließen (MacCormack et al. 2004).

Die Aufspaltung des inzwischen übergroßen Projektes war ebenso sinnvoll, denn jedes Teilprogramm hat schon länger den Umfang eines eigenen Projektes, nicht zuletzt, da plattformübergreifend entwickelt wird. Diese Trennung darf man allerdings nicht missverstehen, denn die Entwickler der Mozilla-Suite[14] arbeiten weiterhin eng zusammen. Die Applikationen bauen teilweise auf den gleichen Komponenten auf und sind mit zahlreichen Extentions erweiterbar, ihre Benutzerführung bleibt einheitlich. Die heiß erwartete Version 1.0 des abgespaltenen Browsers „Mozilla Firefox" erschien im November 2004 und hatte nach nur elf Wochen rekordverdächtige 20 Millionen Downloads zu verzeichnen. Des Öfteren kann man beim Online-Banking Hinweise auf die höhere Sicherheit dieses Browsers im Vergleich zu vorinstallierten Windows–Browsern lesen. Das Mail- und Newsprogramm „Mozilla Thunderbird" landete im

6 SuSE-Linux http://www.novell.com/de-de/linux/suse/
7 Mandrakelinux http://www.mandrakelinux.com/de/
8 Xandros-Desktop http://www.xandros.com/
9 Linspire (http://www.linspire.com) ist eine amerikanische Firma, die ihr Produkt vor einem Rechtsstreit mit Microsoft auch unter dem Namen Lindows anbot. Sie setzt auf kostengünstige Rechner mit vorinstalliertem Linux in Verbindung mit kostenpflichtigen Software-Zusatzpaketen und Support.
10 Freedows http://www.freedows.com ist ein Projekt zweier Firmen aus Brasilien, eine englische oder deutsche Übersetzung ist leider noch nicht verfügbar.
11 IOSN User Guide to Using the Linux Desktop http://www.iosn.net/training/end-user-manual/
12 Linux Client Migration Cookbook http://www.redbooks.ibm.com/abstracts/sg246380.html
13 Video Tutorials http://www.about-linux.com
14 Mozilla Project http://www.mozilla.org/products/

Dezember und wurde innerhalb eines Monats 2 Millionen Mal heruntergeladen. Gelobt werden sein Spam-Filter und die Integration von RSS-Feeds.

Die Mozilla Foundation hat zusammen mit Adobe, Apple, Macromedia, Opera und Sun Microsystems eine neue, offene Plugin-Schnittstelle für Browser angekündigt, die demnächst in Mozilla und Produkten der genannten Partner eingebaut wird (Mozilla Organization 2004). Dies ist ein wichtiger Schritt, um zu verhindern, dass Interaktivität in Webbrowsern auf einzelne Plattformen beschränkt bleibt.

2.4. Basen und Oasen

Distributionen von Linux sind untereinander nicht automatisch kompatibel, andererseits erfordert ihre Pflege aber oft die gleichen oder ähnlichen Arbeitsschritte der Hersteller. Frühere Versuche, eine gemeinsame Basis-Distribution zu erstellen, müssen als gescheitert angesehen werden.[15] Jedoch wagten Conectiva, Mandrakesoft, Progeny und Turbolinux nun kurz nach der Festlegung der Linux Standard Base 2.0[16] einen neuen Anlauf.[17]

Ein noch viel wichtigerer Standard ist der von Dateiformaten. Wer seine privaten oder geschäftlichen Dokumente nicht in einem offenen Format abspeichert, wird immer von der Verfügbarkeit proprietärer Software abhängig bleiben, um darauf zuzugreifen.

Das OpenOffice-Format war von Anfang an offen, jedoch kein allgemeingültiger Standard, bis sich das OASIS-Konsortium dessen angenommen hat und auf Basis des bestehenden OpenOffice-Formats einen neuen Standard für Office-Dateien spezifizierte.[18] Das vor der Tür stehende OpenOffice 2.0 wird ihn unterstützen, natürlich auch kommende Versionen von StarOffice, ebenfalls verwenden wird ihn KOffice 1.4. Die Europäische Union und Sun Microsystems setzen sich dafür ein, dass das Format zum ISO-Standard gemacht wird (Ihlenfeld 2004*b*).

2.5. Dämon beißt in den Apfel

Anders als Linux-Distributionen, die aus GPL-lizenzierter Software bestehen und eher wenige bis gar keine proprietären Zusätze haben, und anders als Windows, das für den Nutzer vollkommen proprietär daherkommt, geht Apple mit seinem MacOSX einen Mittelweg. Der Kernel ist ein FreeBSD-Derivat und wird als OSS-Projekt entwickelt.

Dem System liegen die bekannten und offenen Tools der BSD-Derivate bei, und die sind, wie von BSD-Systemen gewohnt, wie aus einem Guss:

> „Wo sich *Linux* als recht heterogener Baukasten mit diversen Kernel-Varianten und -Versionen, zahlreichen Tools aus unterschiedlichsten Quellen und rund 200 Distributionen mit mehr oder weniger großen

15 Siehe UnitedLinux http://www.unitedlinux.com/.

16 Linux Standard Base http://www.linuxbase.org/

17 Linux Core Consortium, siehe http://componentizedlinux.org/lsb/.

18 OASIS Open Office XML Format Technical Committee http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

> Unterschieden präsentiert, legen die mittlerweile knapp 200 NetBSD–Entwickler (wie ihre Free- und OpenBSD-Kollegen) Wert auf ein homogenes System: Es gibt genau eine aktuelle Distribution, und in der steckt alles drin, was zum System dazugehört." (Diedrich und Kuri 2004)

Damit erbt OSX die Stabilität eines BSDs, umschifft aber die für den typischen Apple-Benutzer zu schwierige Installation, Konfiguration und Wartung. Eine Kommandozeile bekommt man zunächst nicht zu Gesicht. Weiß man aber um sie, kann man von ihr vollen Gebrauch machen. Dies ist zwar auch schon bei einigen Linux–Distributionen so gelöst, jedoch ist durch Apple als zentrales Entwicklungs- und Entscheidungsorgan auch die Oberfläche „Aqua" wie aus einem Guss und bietet die für Apple typische Bedienung. Allerdings erben die Oberfäche und einige Zwischen–Schichten auch die für proprietäre Software typischen Eigenschaften: Anpassungen und Verbesserungen vornehmen kann ebenso wie Fehler und Sicherheitslücken beheben nur der Hersteller, denn er behält die volle Kontrolle über den Code und damit über seine Software. Und ob man mit OSX besser zurechtkommt als mit Windows, KDE oder GNOME, mag jeder für sich entscheiden: Die Benutzbarkeit eines Systems ist zu einem großen Teil auch vom individuellen Geschmack und der persönlichen Gewöhnung des Benutzers abhängig.

2.6. Open-Source-Usability

Der erste Artikel des Technik-Kapitels beschäftigt sich mit der Usability (Benutzungsfreundlichkeit) von Open-Source-Software. Er trifft genau ins Schwarze, um ihre charakteristischen Stärken und Schwächen für den Einsatz auf dem Desktop auszuloten. Grundlage für den Artikel sind eine Menge praktischer Erfahrungen mit der Materie, unterfüttert von einer Usability-Studie. Autor des Artikels ist Jan Mühlig, Gründer und Vorstand der relevantive AG, einem IT-Unternehmen aus Berlin, das insbesondere der KDE-Community nahe steht.[19] Neben der Mitarbeit im KDE–Projekt und einer Usability-Kolumne im Magazin „LinuxUser" hat das Team von relevantive den Usabiliy-Track auf der KDE-Konferenz aKademy 2004 maßgeblich gestaltet und ist somit bestens qualifiziert, Erfahrungen aus erster Hand mitzuteilen.

3. Freie Software-Entwicklung

Kein Thema ist so nah dran an Open Source wie seine Entwicklung, denn man kann mit dem Begriff „Open Source" nicht mehr nur eine Gruppe von Lizenzen bezeichnen: Mit der Community hat sich eine ganz neue Art der Software-Entwicklung etabliert. Auch wenn viele OSS-Projekte leichte Unterschiede im Entwicklungsprozess aufweisen, so haben sie alle doch eine gemeinsame Grundlage: den ungehinderten Austausch von Code.

19 Zum Team von relevantive gehört noch Ellen Reitmayr. An der Linux-Usability-Studie haben außerdem Eva Brucherseifer und Ralf Ackermann von basysKom und Jutta Horstmann mitgewirkt.

3.1. Blick über den Tellerrand

Ein sehr interessantes Thema, dessen Beobachtung sich über die nächsten Jahre hinweg sicher lohnen wird, ist das der APIs. Ein *Application Programming Interface* ist sicherlich die wichtigste Schnittstelle für einen Programmierer und kann indirekt entscheidend sein für den Erfolg und die Verbreitung eines Softwaresystems und ist damit auch ein wichtiges Instrument im Konkurrenzkampf für Closed-Source-Giganten wie Microsoft und Apple. Ein ehemaliger Microsoft-Mitarbeiter und Gründer einer Software-Firma erklärt in seinem Artikel, warum er davon ausgeht, dass Microsoft den „API War" bereits verloren hat (Spolsky 2004). Er führt dies auf die radikale Einführung neuer Windows-APIs zurück, die in zukünftigen Windows-Versionen keine Rückwärts-Kompatibilität mehr liefern werden. Obwohl die neuen Schnittstellen technisch durchaus gut entworfen seien, fingen die Programmierer in Softwarefirmen mehr oder weniger bei null an und müssten sich zudem mit zwei konkurrierenden Grafik-APIs herumschlagen. Programmierer mit der entsprechenden Expertise seien entsprechend knapp verfügbar und trieben die Personalkosten nach oben.

Beim Nennen des Begriffs *Registry* gehen bei vielen Menschen haufenweise rote Lichter an. Der Grund ist die schlechte Implementierung der *Registry* unter Microsofts Windows, die ihren Ruf als Geschwindigkeitsbremse und Wartungsproblem verdient hat. Dennoch hat es sich das Projekt Elektra[20] zum Ziel gesetzt, eine *Registry* für Linux zu entwerfen und zu implementieren, allerdings ohne die bekannten Design–Schwächen dabei nachzubauen.

Während Longhorn noch lange auf sich warten lässt, ist 64-Bit-Computing in der Open-Source-Welt bereits Realität. Mono 1.0 ist im Juni 2004 angekommen und hat einen wichtigen Grundstein dafür gelegt, dass dot.NET-Applikationen auch unter Linux laufen werden. Schon länger verbreitet sind Java-Applikationen, die auch unter Linux ihren Dienst wie gewohnt verrichten,[21] trotzdem Sun sich auch 2004 nicht entschließen konnte, sein Java unter eine echte Open-Source-Lizenz zu stellen. Sun verbreitet den Code zwar, lässt Modifikationen aber nicht frei zu.[22] Immerhin wird es von Suns Betriebssystem Solaris demnächst eine offene Version namens „OpenSolaris"[23] geben, die eine interessante Alternative zu Linux und *BSD werden könnte. Besonders bemerkenswert ist die angestrebte Kompatibilität zu Linux-Applikationen, vorerst aber sollen wichtige Teile wie der Kernel nicht unter eine Lizenz gestellt werden, die eine Weitergabe von Modifikationen erlaubt (Ziegler 2004).

3.2. „BUGS!"

Fehler im Code sind jedem Programmierer ein Greuel. Im Jargon werden sie Bugs genannt. Kurz nach Anfang der (damals noch langsameren) computertechnischen Geschichtsschreibung finden wir einen kleinen Störfall, der sich 1945 während eines

20 Elektra Initiative http://elektra.sourceforge.net/

21 Java Linux http://www.blackdown.org/

22 Eine freie Java-Implementation im Sinne der GPL wird derzeit im GNU-Projekt Classpath erstellt, (http://www.gnu.org/software/classpath/).

23 OpenSolaris http://www.opensolaris.org/

Testlaufs des Urgesteins Mark II an der Harvard University ereignete (Naval Historical Center 1999). Eine Motte verfing sich in einem der zahlreichen Relays. Das von Grace Hopper in das Logbuch geklebte Insekt kann nicht nur im Museum bestaunt werden, sondern gilt als der Ursprung der Bezeichnung Bug für Fehler bzw. der Tätigkeit des *Debugging*, bei der man Fehler in Programmen aufstöbert und sie entfernt.

In Open-Source-Software können Bugs leicht gefunden und beseitigt werden. Durch den offenen Programmcode ist diese Eigenschaft implizit immer vorhanden, eine Garantie, dass diese Möglichkeit auch genutzt wird gibt es aber nicht. Eine Belohnung[24] für sachdienliche Hinweise zum Aufspüren von Fehlern weckt bunte Assoziationen, angefangen bei den „Wanted"-Steckbriefen im Wilden Westen, über Kopfgeldjäger jenseits des Amazonas bis hin zu den Fahndungsplakaten des BKA. Diese Methode ist jedoch nicht für alle OSS-Projekte gleichermaßen geeignet.

3.3. Kern der Dinge

Die vermutlich bekannteste freie Software ist der Linux-Kernel. Seine Entwicklungsmethode stand in der Vergangenheit immer wieder in der Kritik, da auch in den „stabilen" Kernel-Versionen bereits schwerwiegende Änderungen vorgenommen und damit auch teils schwere Fehler eingebaut wurden. Linus Torvalds (2004*b*) selbst stellte in einem längeren Text einige Dinge über seinen Management-Stil als Entwicklungsleiter klar und ging auch ausführlich auf den persönlichen Umgangston ein, der nicht rüde werden dürfe: „don't call people d*ckheads (at least not in public)". Beim Kernel Summit 2004, dem traditionellen Treffen der Linux-Kernel-Entwickler, wurden die Änderungsmöglichkeiten erörtert. Die Diskussion verlief überwiegend zugunsten gradueller Änderungen und paralleler Test-Kernels (LWN.net 2004).

Der Finne Ilkka Tuomi analysiert in seinem Artikel „Evolution of the Linux Credits" (2004) die Entwickler-Gemeinde des Linux-Kernels. Er beschreibt dabei eine automatische Auswertung der Datei „CREDITS", die sich als schwierig entpuppt aufgrund der teils sehr unterschiedlichen Weise, wie die vorgesehenen Felder (z. B. Beschreibung oder Adresse) ausgefüllt wurden.

Die Datei war gedacht, um die Teilnahme am Kernel-Projekt festzuhalten, jedoch eignet sie sich nicht für einen juristisch verwendbaren Nachweis über die Urheberschaft bestimmter Code-Teile. Deshalb und aus gegebenem Anlass – die Diebstahlsbehauptungen von SCO standen im Raum – wurde das „Signieren" von Patches [25] eingeführt, damit kein Code aus unbekannten Quellen in den Kernel aus Versehen oder absichtlich eingeschleust wird (Torvalds 2004*a*).

Vor allem über Anzahl und Wohnsitz lassen sich dennoch Daten aus dem Credits–File gewinnen: So waren bis zur Linux-Version 2.5.25 Mitte 2002 über 400 Personen

24 Mozilla Foundation Announces Security Bug Bounty Program http://www.mozilla.org/press/mozilla-2004-08-02.html

25 Der „unterzeichnende" Entwickler bestätigt mit einem „Developer's Certificate of Origin", dass der Code von ihm selbst oder einem ihm bekannten Entwickler unter einer Open-Source-Lizenz geschrieben wurde oder auf einem solch lizenzierten Code aufbaut.

am Projekt beteiligt, davon 154 aus den Vereinigten Staaten und 69 aus Deutschland. Dahinter lagen Großbritannien, Kanada, Australien und die Niederlande mit 32–18 Entwicklern. Nimmt man die Anzahl aus allen EU-Ländern zusammen, so überstieg diese mit 187 sogar die Zahl der amerikanischen Entwickler. Auch wenn in vielen anderen Staaten meistens weniger Entwickler beheimatet sind, so ist Linux dennoch ein sehr internationales Projekt, das – vereinfacht gesagt – alle etwaigen Probleme überwunden hat, von nur einem einzigen Entwickler auf über 400 Beitragende zu skalieren.

3.4. Software-Evolution

Mit dem Thema der Software-Entwicklung beschäftigt sich der zweite Artikel des Technik-Kapitels: Markus Pizka und Andreas Bauer von der Technischen Universität München haben sich die Mechanismen der Open-Source-Entwicklung einmal genauer angesehen und erklären, warum sie oft besser skalieren und funktionieren als unter den Bedingungen kommerzieller Entwicklung von Closed-Source-Software.[26] Bei der Software-Evolution geht es um die kontinuierliche und kontrollierte Weiterentwicklung von Software und insbesondere um die Veränderung des Entwicklungsprozesses an sich.[27]

4. Open Source im Jahrbuch

Es ist nur konsequent, wenn in einem Buch über Open Source auch Open Source drin ist. Richtig, gemeint ist Programmcode, den man lesen und mit ausreichend fachlicher Kenntnis auch verstehen kann. Wer nicht selber programmiert, hat vielleicht noch nie gesehen, wie die „Kochrezepte" der Informationstechnologie aussehen. Wer nicht allerengste Beziehungen zu einer „Softwareschmiede" pflegt, für den ist Open Source die einfachste Möglichkeit, sich einen Eindruck zu verschaffen.

Im Rahmen dieses Buches können wir natürlich nur ein kleines Programm stellvertretend für die Massen an offenem Sourcecode abdrucken, die täglich geschrieben, verbessert und verwendet werden. Um die Auswahl aus dieser schier unendlichen Fülle zu erleichtern, haben wir, die Redaktion des Jahrbuchs, im Herbst 2004 einen Wettbewerb ausgerufen. Aus den Einsendungen wurde auf unserer Internetseite[28] das Programm Snowbox von Oliver Feiler zum Gewinner gewählt. Gratulation!

Oliver Feiler hat für seinen mit OpenBSD betriebenen Server einen einfachen und sicheren POP3-Server gesucht. Nachdem er mit bereits vorhandenen Lösungen nicht zufrieden war, hat er sich dazu entschlossen, selber einen zu schreiben. Sein eher minimalistischer Ansatz legt den Schwerpunkt auf wenig Code, Sicherheit und einfache Installation und Konfiguration. Er hat Snowbox in Perl geschrieben:

26 Einen flüchtigen, aber interessanten Einblick in die proprietäre Softwareproduktion gibt z. B. auch das Blog von Rick Schaut (http://blogs.msdn.com/rick_schaut/archive/2004/05/19/135315.aspx).

27 Nicht gemeint sind genetische Algorithmen, siehe dazu „Linux: Tuning The Kernel With A Genetic Algorithm" http://kerneltrap.org/node/4493.

28 Open-Source-Jahrbuch http://www.Think-Ahead.Org

> „Perl, weil es sehr einfach ist, damit Text zu verarbeiten, und weil es die Kriterien 'so wenig Code wie möglich' und 'so sicher wie möglich' am ehesten erfüllt. Ein Server sollte vor allem erstmal sicher sein, an der Geschwindigkeit kann man immer noch drehen. Und da es unmöglich ist, mit Perl klassische Buffer Overflows, wie wir sie alle von C 'lieben', zu programmieren und ich Perl recht gut kenne, fiel die Wahl darauf. Darüber hinaus sieht man mit Perl recht schnell Resultate. Die erste funktionierende Version war bereits nach wenigen Stunden fertig." (Oliver Feiler)

Das Programm hat er unter die GPL gestellt, da diese Lizenz seinen Vorstellungen entspricht und weithin bekannt ist. Aus „Snow" – er liebt Schnee und Schneeleoparden – und „box" vom englischen Wort Mailbox für Briefkasten wurde schließlich der Name Snowbox.[29]

5. Eingebettete Systeme

Von digitalen Assistenten, die bald in unsere Kleidung eingearbeitet sein könnten, über Bordcomputer für Autos oder Kühlschränke, bis zur Technik, die die International Space Station im Orbit hält, besteht zunehmend Bedarf an eingebetteten Systemen.

> „I get the biggest enjoyment from the random and unexpected places. Linux on cellphones or refrigerators, just because it's so not what I envisioned it." (Linus Torvalds, zitiert bei Dudley 2004)

Eine primäre Aufgabe des Betriebssystems ist die Verwaltung von Hardware-Ressourcen. Unabdingbar sind dazu Treiber, mit denen die Hardware angesteuert werden kann.

5.1. Linux-Treiber

Bei Linux sind sie ein heißes Eisen. Die Treiber sind zumeist fest im Kernel eingebaut, der Vorteil ist eine sehr gute Integration der Treiber in das System. Der Nachteil ist, dass Treiber nicht beliebig hinzugeladen werden können. Zwar können ausgewählte Teile des Kernels als Kernelmodule dynamisch geladen werden, diese müssen aber bereits in die Quellen des Kernels eingebettet sein.

Die Integration erfordert die Offenlegung der Treiber-Quellen, ein Schritt, den viele Hersteller nicht gehen wollen, da sie die Funktionsweisen ihrer Geräte oft als Geschäftsgeheimnis ansehen oder nicht gehen können, da Teile ihres Treibers lizenzierten Code von Dritten enthalten. Eine Möglichkeit, dies zu umgehen, sind binäre Kernel-Module, bei denen nur die Einbindung in den Kernel offen liegt. Genau dies ist die Vorgehensweise von NVidia bei den Linux-Treibern ihrer Grafikhardware.

29 Aktuelle Versionen von Snowbox und andere Software von Oliver Feiler finden sich auf seiner Homepage http://kiza.kcore.de/software/ oder bei Freshmeat http://freshmeat.net/~kiza/.

Binäre Kernelmodule werden in Teilen der Open-Source-Szene nicht gern gesehen, aber ohne diese Lösung gäbe es vermutlich noch weniger Treiber.

„Pranger für Hardwarehersteller?" (Lindner 2004*b*) lautete eine provokante Schlagzeile: Als Benutzer von Nischensystemen kann man sich nämlich sehr leicht im Stich gelassen fühlen, wenn die auserwählte Plattform von großen Hardware-Herstellern systematisch übergangen wird. Unverständnis macht sich breit, denn der Benutzer kann nicht verstehen, warum Hersteller freiwillig auf Kunden verzichten. Dies gilt ganz besonders für Betriebssysteme wie Linux, die inzwischen zum „Mainstream" gehören (IDC 2004). Die teils desolate Treibersituation bei gängiger Hardware und Peripherie erscheint manchmal absurd und hindert vor allem unbedarfte Privatnutzer am dauerhaften Wechsel zu Linux.

Ein negatives Beispiel ist die WLAN-Unterstützung auf Hardware von Intel und Apple. Es gab zum Erscheinungstermin von Intels Centrino-Komponenten für Notebooks zunächst nur einen Windows-Treiber. Für Linux wurde ein Treiber zwar angekündigt, dann aber auf Eis gelegt. Erst weit über ein Jahr später steht nun ein erster nativer Treiber für eine inzwischen ältere Bauart der Centrino-Chips zur Verfügung, der in Zusammenarbeit mit einem Open-Source-Projekt entwickelt wurde (Lindner 2004*a*). Wer sich einen neueren Schoß-Rechner von Apple zugelegt hat, kann Airport Extreme[30] auch weiterhin nicht unter Linux oder freien BSDs nutzen, ein Treiber oder die Freigabe von Spezifikationen ist nicht geplant. Als kleiner Trost bleibt den üblichen Verdächtigen mit ihren schnuckeligen iBooks nur, dass unter der Haube ihres MacOSX ohnehin schon ein Unix werkelt.

Vor unserem kleinen Treiber-Exkurs wurden einige Ideen für eingebettete Systeme bereits erwähnt, einige konkrete Beispiele und Entwicklungen sollen dem nun folgen.

5.2. Beispiele für OSS auf eingebetteten Systemen

Obwohl die Zahl der mit Linux betriebenen PDAs leider gering ist, scheint sich im restlichen *Embedded*-Bereich eine ganz andere Entwicklung abzuzeichnen. Dort ist der Anteil von Linux-Geräten inzwischen so hoch, dass Linux als erste Wahl für den *Embedded*-Bereich gelten kann (LinuxDevices.com 2004).

So läuft Linux inzwischen nicht nur auf Handys[31] und Navigationssystemen, es gibt auch ganze Linux-Computer, die komplett auf eine CompactFlash-Karte passen. LinuxDevices.com ist wohl das bekannteste und umfassendste Portal für eingebettete Systeme mit Linux. Von der Fernbedienung bis zum Router finden sich dort fast alle mit Linux betriebenen Geräte.

Da eingebetteten Systemen meistens keine großen Hardware-Ressourcen zur Verfügung stehen, macht es Sinn, den Kernel und die verwendeten Tools daran anzupassen und nicht benötigte Treiber und Programme zu entfernen. So gibt es jetzt z. B. auch

30 Das Airport-Extreme-Modul verwendet einen 802.11g-Chipsatz von Broadcom.

31 Motorola richtet den Mittelpunkt seiner Mobiltelefon-Strategie auf Linux aus: „There's a need to move to a more open, flexible architecture for the majority of handsets and [to provide] a more open developer platform [. . .] The great thing [about Linux] is that some of the brightest minds are helping develop it" (Andrew Till, zitiert bei Computer Business Review Online 2005).

von der Linux-Distribution Gentoo einen Ableger für eingebettete Systeme.[32] Eine auf Sicherheit ausgerichtete *Embedded*-Distribution ist AMSEL.[33]

Unter den bereits angesprochenen Problemen mit nicht vorhandenen Treibern leidet nicht nur Linux, auch die BSDs sind betroffen. Das bewog das OpenBSD-Projekt, eine Kampagne für die freie Bereitstellung von Firmware zu starten, da die benötigten Laderoutinen für die Firmware vieler Geräte oft in geschlossenen Windows-Treibern integriert sind (Andrews 2004).

Die OpenIB-Allianz hat sich zum Ziel gesetzt, einen performanten und skalierbaren Linux-Stack für die InfiniBand-Architektur[34] zu entwickeln. Dafür hat sie im November 2004 eine Förderung vom amerikanischen Energieministerium erhalten (HPCwire 2004). Für diese Zukunftstechnologie wird Linux also gut gerüstet sein:

> „the embedded market space [. . .] may not get as much notice, and might not be as in-your-face as the desktop, but it's clearly a growing area for Linux, too." (Linus Torvalds, zitiert bei Dudley 2004)

5.3. Embedded Linux Development

Der Beitrag von Joachim Henkel und Mark Tins über die offene Entwicklung bei kommerziellen Herstellern von Systemen mit eingebettetem Linux schließt unseren technisch angehauchten Artikel-Reigen und bildet einen guten Anschluss an das Kapitel „Ökonomie". Die Autoren zeigen u. a. auf, dass Hersteller von Hardware unter bestimmten Marktbedingungen ihre Programmquellen doch offen legen können, ohne Nachteile im Wettbewerb befürchten zu müssen, sondern von dieser Offenlegung sogar noch profitieren werden.

Literatur

Andrews, J. (2004), 'Feature: OpenBSD Works To Open Wireless Chipsets'. http://kerneltrap.org/node/4118.

Computer Business Review Online (2005), 'Motorola promises greater Linux phone focus', Artikel vom 21. Januar, http://www.cbronline.com/article_news.asp?guid=65BFD343-7E33-4DB0-8F3E-F7F2E6EAF712.

Diedrich, O. und Kuri, J. (2004), 'Das Allerwelts-Unix: 10 Jahre NetBSD', heise online http://www.heise.de/newsticker/meldung/35544/.

Dudley, B. (2004), 'Q&A: Linus Torvalds, inventor of Linux', The Seattle Times http://seattletimes.nwsource.com/cgi-bin/PrintStory.pl?document_id=2002059632&zsection_id=268448455&slug=linus11&date=20041011.

HPCwire (2004), 'OpenIB Alliance Members Win DOE Grant', **13**(45B). LIVEwire Edition http://www.tgc.com/hpcwire/hpcwireWWW/04/1110/108788.html.

32 Gentoo Embedded http://www.gentoo.org/proj/en/base/embedded/index.xml

33 AMSEL (Advanced Modular Secure Embedded Linux) http://www.amselinux.de/

34 InfiniBand http://infiniband.sourceforge.net/

IDC (2004), 'The Linux Marketplace – Moving From Niche to Mainstream', http://www.osdl.org/docs/linux_market_overview.pdf. OSDL Pressemitteilung: http://www.osdl.org/newsroom/press_releases/2004/2004_12_15_beaverton.html.

Ihlenfeld, J. (2004*a*), 'Interview: NX – die Revolution des Netzwerk-Computing?', Golem IT-News http://www.golem.de/0408/33026.html.

Ihlenfeld, J. (2004*b*), 'Setzt die EU künftig auf Dateien im OpenOffice.org-Format?', Golem IT-News http://www.golem.de/0409/33806.html.

LWN.net (2004), 'Kernel Summit: Development process', LWN.net http://lwn.net/Articles/94386/.

Lindner, M. (2004*a*), 'Intel Centrino Treiber 1.0.0', Pro-Linux http://www.pro-linux.de/news/2004/7502.html.

Lindner, M. (2004*b*), 'Pranger für Hardware-Hersteller?', Pro-Linux http://www.pro-linux.de/news/2004/7203.html.

Lindner, M. (2005), 'Das Linux-Jahr 2004 – Ein kleiner Rückblick', Pro-Linux http://www.pro-linux.de/berichte/rueckblick2004.html.

LinuxDevices.com (2004), 'Linux Now Top Choice Of Embedded Developers', http://www.linuxdevices.com/news/NS2744182736.html.

MacCormack, A., Rusnak, J. und Baldwin, C. (2004), 'Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code', Harvard Business School Working Paper Number 05–016, http://opensource.mit.edu/papers/maccormackrusnakbaldwin.pdf.

Mozilla Organization (2004), 'Mozilla Foundation Announces More Open, Scriptable Plugins', Pressemitteilung http://www.mozilla.org/press/mozilla-2004-06-30.html.

Naval Historical Center (1999), 'Rear Admiral Grace Murray Hopper, USNR, (1906–1992)', http://www.history.navy.mil/photos/pers-us/uspers-h/g-hoppr.htm. Department of the NAVY, Photo #:NH 96566-KN (Color).

Spolsky, J. (2004), 'How Microsoft Lost the API War', http://www.joelonsoftware.com/articles/APIWar.html.

Torvalds, L. (2004*a*), 'Linus on documenting patch provenance', LWN.net http://lwn.net/Articles/86436/.

Torvalds, L. (2004*b*), 'Linus on kernel management style', LWN.net http://lwn.net/Articles/105375/.

Tuomi, I. (2004), 'Evolution of the Linux Credits file: Methodological challenges and reference data for Open Source research', *First Monday* **9**(6). http://firstmonday.org/issues/issue9_6/tuomi/index.html.

Ziegler, P.-M. (2004), 'Start von OpenSolaris verzögert sich', heise online http://www.heise.de/newsticker/meldung/52953/.

Open Source und Usability

JAN MÜHLIG

Open-Source-Software (OSS) steht im Ruf, wenig nutzungsfreundlich und daher für „normale" Anwender kaum geeignet zu sein. Einige selbsternannte „Beobachter" führen dies vor allem auf das geringe Interesse der nicht–kommerziellen Entwickler zurück, sich mit Ergonomie oder Usability (Nutzungsfreundlichkeit) zu beschäftigen. Meine Erfahrungen mit verschiedenen Open-Source-Projekten führen zu einem differenzierteren Bild. In meinem Aufsatz gehe ich zunächst auf einige Strukturbedingungen der OSS-Entwicklung ein und zeige, wie sie sich auf Usability auswirken – soweit möglich im Vergleich zu kommerzieller Software-Entwicklung. Im Weiteren beschreibe ich die Herausforderungen, wie in Zukunft Benutzungsfreundlichkeit im Open-Source-Bereich umgesetzt und fest etabliert werden kann. Hier spielen zwei Faktoren eine wesentliche Rolle: Die Integration von Usability und Ergonomie in die Praxis der Open-Source-Entwicklung sowie die Bereitstellung und Schulung von Usability-Ressourcen (Experten, Einrichtungen, Wissen). Wenn dies umgesetzt werden kann, dann hat OSS mittelfristig die Chance, Benutzerfreundlichkeit als (Markt-)Vorteil auszuspielen.

1. Einleitung

Der Erfolg und die Verbreitung von Open-Source-Software (OSS) in den vergangenen Jahren gründen sich vor allem auf dem Server- und Backend-Bereich. Auf dem Desktop ist der Erfolg bislang begrenzt. Entscheidungen, wie die der Stadt München zur Migration mit Linux als Desktop-System, verleiten zu der Annahme, dass es nur noch eine Frage der Zeit ist, bis OSS seine Erfolgsgeschichte auch auf dem Desktop fortführt.

Um im Desktop-Bereich mittel- und langfristig konkurrieren zu können, muss OSS benutzerfreundlich sein, und zwar auch für „normale Anwender". Eine Software kann für den Anwender nur soweit nützlich und sinnvoll sein, wie er sie benutzen kann. Die Frage stellt sich also, wie benutzbar bzw. benutzerfreundlich OSS ist. Tatsächlich gibt es bislang kaum vergleichende Untersuchungen, und die von uns, der relevantive AG, im Frühjahr 2003 durchgeführte Usability-Studie (Mühlig et al. 2003) zu Linux[1] im

1 SuSE-Distribution mit KDE-Arbeitsumgebung

administrierten Desktop-Bereich war eine der ersten, die Daten dazu lieferte.[2] Seitdem sind leider kaum weitere Studien dieser Art dazugekommen.

Ebenfalls ist bislang wenig darüber bekannt, welche Rolle Usability in OSS-Projekten spielt. Pionier-Arbeit dazu leisteten Nichols und Twidale (2002) mit ihrem Aufsatz „Usability and Open Source Software", der allerdings vor allem aus den strukturellen Bedingungen Schlüsse zieht. Empirische Erfahrungen sind dagegen noch immer selten.

In diesem Aufsatz werde ich einige der Faktoren, die für oder gegen die Entwicklung von benutzerfreundlicher Software im Open-Source-Bereich sprechen, aufgreifen und – soweit möglich – mit meinen Erfahrungen aus verschiedenen Projekten, insbesondere dem KDE-Projekt, gegenüberstellen.

2. Entwickeln für die Community

Eines der häufigsten Argumente, warum OSS wenig benutzerfreundlich sei – in der Regel ohne konkreten Beleg – ist die traditionelle Ausrichtung an der Community, d. h. an anderen Hackern (Nielsen 2004). Ihr Urteil ist maßgebend für die Qualität einer Software, entsprechend wird sie mehr oder weniger für diese Gruppe als Endanwender geschrieben. Andere Projekte sind daraus entstanden, dass man eine Software entwickelte, die ziemlich genau das tut, was man selbst gerne haben möchte. Kann man dann verlangen, dass diese Software auch für „normale Anwender" benutzerfreundlich sein muss? „Wer andere Bedürfnisse hat, kann ja eine andere Software einsetzen" ist ein häufig gehörtes und durchaus plausibles Argument.

Diese Einstellung wird jedoch fragwürdig, wenn man Interesse daran hat, dass die eigene Software auch die Wahl für einen Behörden-Desktop oder Windows-Umsteiger ist. Solche Nutzer haben andere Ansprüche als die Entwickler. Deren Rationalität zielt nicht darauf herzustellen, was man möchte, sondern einzusetzen, was man braucht. Will man also Software für durchschnittliche Nutzer einfacher benutzbar machen, dann kann man sie nicht nur für sich selbst schreiben.

Damit tut sich eine sehr wichtige Konfliktlinie auf: Programmiert man für sich und andere OSS-Entwickler oder für „durchschnittliche" Nutzer? Dass sich beides häufig ausschließt, zeigt sich z. B. bei der Funktionsvielfalt. Die meiste OSS hat einen reichen Schatz an Funktionalität, der natürlich für die speziellen Bedürfnisse der Entwickler durchaus passt. So gibt es im Mailprogramm KMail die Einstellung, dass ein Ordner eine Mailingliste beinhalten kann. Diese Funktion ist praktisch für die Verwaltung von Mailinglisten. Ein Entwickler fand dieses Feature sinnvoll, er programmierte es, und es wurde in die Software aufgenommen. Die Zahl derer außerhalb der Hacker-Community, die dieses Feature verwenden, dürfte dagegen sehr klein sein. Trotzdem steht es auf einer Ebene mit z. B. Ordner-Namen oder Icons. Für den Nutzer wird diese Funktionsvielfalt schwierig, weil er nicht sofort unterscheiden kann, was wichtig für

2 Die Studie untersucht, wie leicht typische Aufgaben im beruflichen Einsatz unter Linux bzw. Windows XP erledigt werden können. Dabei wurden 60 (an Linux) bzw. 20 (an Windows XP) Personen getestet. Der Ergebnis-Report ist kostenlos unter http://www.relevantive.de/Linux-Usabilitystudie.html verfügbar.

ihn ist, und was nicht. Heißt das aber, dass man nur noch jene Funktionen anbietet, die für den „normalen" Nutzer relevant sind? Auf welche Funktionen verzichtet man? Was sagen jene Entwickler, die diese Funktion programmiert haben? Was sagen jene Nutzer, die bisher die Software genau deswegen einsetzen, weil es diese Funktion gibt? Diese Konflikte traten z. B. auf, als ich Menü-Dialoge von KMail konzeptionell überarbeitete, um sie leichter benutzbar zu machen. Die „Orchideen"-Features herauszunehmen oder zumindest zu verstecken, würde die Übersichtlichkeit verbessern und die schnelle Erfassung der zentralen Funktionen erhöhen. Auf der anderen Seite stößt man Entwickler vor den Kopf. Ob sich ein Projekt dafür entscheidet, die eigenen Wünsche und die der Community zurückzustellen, um „massenkompatibler" zu werden wird sicherlich in Zukunft an Bedeutung zunehmen. Vielleicht sind im Einzelfall Kompromisse möglich, mit denen es sich leben lässt. Doch ganz auflösen lässt sich dieser Konflikt nicht.

3. Usability ist einfach?

In einem sehr kontrovers diskutierten Aufsatz lässt sich der wichtige OSS-Vorkämpfer Raymond (2004) über die mangelhafte Benutzbarkeit der CUPS-Druckerkonfiguration aus, die es selbst ihm sehr schwer machte, einen Netzwerkdrucker in Betrieb zu nehmen. Raymond greift die Entwickler für ihre Nachlässigkeit an und resümiert, sie müssten nur „Tante Tillie"[3] vor Augen haben, dann würden sie die Software schon so gestalten, dass erstere sie leicht bedienen kann: Usability sei einfach.

Diese Einstellung ist sehr weit verbreitet, auch in der kommerziellen Softwareentwicklung. Tatsächlich wird dabei ein grundsätzliches Missverständnis darüber, was Usability bedeutet, sichtbar, denn: Woher weiß ich, was Tante Tillie braucht? Habe ich sie gefragt? Habe ich sie dabei beobachtet, wie sie eine Software einsetzt? Weiß ich, warum sie etwas so und so tut und nicht anders? Weiß ich, wie sie welche Begriffe versteht? Vermutlich nicht. Stattdessen wird eine Vorstellung von der Nutzungswelt einer fiktiven Tante erstellt, die zudem das „untere Ende" aller gewünschten Nutzer darstellen soll. Gruber (2004) weist vollkommen zurecht auf die Geringschätzung für die „dumb users" hin, die daraus spricht. OSS scheint für eine solche Haltung empfänglicher zu sein, weil sie ihren Erfolg im Zweifelsfall nicht von einem „Markt" abhängig machen muss, wie dies für kommerzielle Software der Fall ist.

Usability ist (leider) nicht so trivial, als dass man nur an den richtigen Nutzer denken müsste. Wäre das der Fall, dann würden wir vermutlich in einer Welt leben, in der Software „verschwindet", weil sie sich so nahtlos in unsere Bedürfnisse einfügt, dass wir sie gar nicht mehr wahrnehmen. Usability ist aber ein durchaus aufwändiger Prozess, im Verlauf dessen die Software *so* an die Nutzer angepasst wird, dass sie intuitiv, erfolgreich, effizient und angenehm anzuwenden ist. In der klassischen Softwareentwicklung wird dieses Ziel dadurch zu erreichen versucht, dass man die Anforderungen der zukünftigen Nutzer erhebt und bestimmt (Requirements), dann schrittweise die Software entwickelt und z. B. durch Prototypen wiederholt an der

3 Gemeint ist der berühmte „dümmste anzunehmende User" kurz DAU.

„Wirklichkeit", also Nutzern, überprüft. Dieses Verfahren ist aufwändig und teuer, benötigt Usability-Spezialisten und muss nicht zuletzt vom Projektmanagement gewollt sein. Andernfalls ist es auch in kommerzieller Softwareentwicklung lediglich ein Lippenbekenntnis. OSS-Projekte können für sich entscheiden, ob sie für durchschnittliche Nutzer geeignet sein sollen oder nicht. Entscheiden sie sich dafür, müssen sie genauso Verfahren entwickeln und Ressourcen aufbauen, um dieses Ziel zu erreichen.

4. OSS-Usability = Bazaar?

Das Mitwirken in OSS-Projekten wurde von Raymond (1998) in seinem berühmten Aufsatz „The Cathedral and the Bazaar" mit den Mechanismen eines Bazars beschrieben. OSS-Entwicklung zeichnet sich durch offene Kommunikationsstrukturen aus: Es ist leicht, mit den Entwicklern in Kontakt zu treten und Feedback einzubringen. Eigentlich müsste die Open-Source-Community sehr gute Voraussetzungen dafür haben, durch einen Austausch mit den Nutzern zu benutzerfreundlicher Software zu gelangen. Kann Usability-Kontribution gleichermaßen funktionieren? Gibt es wesentliche Unterschiede zwischen Usability-Kontribution und Code-Entwicklung, die dagegensprechen?

Die Maintainer eines Projektes können hervorragend abschätzen, was guter Code ist und was nicht, auch wenn sie den Entwickler nicht kennen. Das gleiche gilt für Bugs. Sie sind (im Regelfall) objektiv reproduzierbar und entweder tatsächlich ein Bug oder eben nicht.

Das sieht bei Usability-Beiträgen anders aus. Angenommen, jemand schickt den Maintainern des Projektes X einen Problembericht und einen Vorschlag zur Verbesserung. Woher wissen die Maintainer, dass es sich tatsächlich um ein Problem handelt? Woher wissen sie, dass der Vorschlag eine Verbesserung ist? Für wen?

Usability-Kontribution ist daher vermutlich nicht einfach mit Code vergleichbar. Die einzige wirklich seriöse Grundlage für Aussagen sind Tests oder Beobachtungen mit „echten" Nutzern. Auch Aussagen, die aufgrund von früheren Tests oder vergleichbaren, verifizierten Situationen stammen, sind ausreichend, wenn sie vorsichtig angewendet werden. Relativ einfach ist schließlich die Überprüfung auf Konsistenz oder Norm- bzw. Guidelines-Konformität („heuristic evaluation").

Ohne zumindest eine dieser Grundlagen ist Usability reine Spekulation oder schlicht Wichtigmacherei. Wenn also – wie in vielen Foren und Mailinglisten der Fall – kein Teilnehmer der Community diese Basis hat, kann das Ziel einer verbesserten Benutzbarkeit für „durchschnittliche Nutzer" kaum erreicht werden. Leider herrschen dort meist persönliche Meinungen oder Projektionen vor.

Daher erscheint es mir gegenwärtig unwahrscheinlich, dass Usability im OSS–Bereich das gleiche „Wunder" des Bazars erfährt, bei dem alle gleichzeitig sprechen und auf magische Weise das Richtige herauskommt. Wohl ist eher das Gegenteil der Fall. In der Vielzahl der Stimmen kann der Maintainer kaum entscheiden, was er nun berücksichtigen soll und was nicht. Dieser Umstand kann sich erst dann ändern, wenn wirklich auf Grundlage von Tatsachen argumentiert wird und diese tatsachenbasierten

Diskussionen eine kritische Masse erreicht haben, auch um einen Standard zu setzen. Gegenwärtig ist dieser Stand nicht erreicht.

Kommerzielle Software hat hier andere Voraussetzungen: Die Entscheidung, welche Probleme repräsentativ für die Zielgruppe sind, wird von Usability-Experten übernommen, nicht von Entwicklern. Aufgrund von Aufwandsabschätzungen durch die Entwickler entscheidet das Produktmanagement über die Umsetzung.

Die klassischen Feedback-Kanäle sind darüber hinaus gerade für „normale" Anwender schwierig zu benutzen: Sie sind mit IRC und Bug-Tracking-Systemen auf technisch versierte Nutzer ausgerichtet, filtern jene, die diese Voraussetzung nicht erfüllen, schlichtweg aus. Doch selbst wenn hier einfache Schnittstellen zwischen Anwendern und Entwicklern geschaffen würden, so wären sie kaum sinnvoll für die OSS-Projekte nutzbar. Wenn tausende von Nutzern „ihr Problem" oder „ihre Lösung" einreichen, bliebe kaum mehr Zeit zum Coden. Die Wirkung wäre einer DDoS-Attacke vergleichbar. Und das Problem des „wer hat jetzt Recht" bleibt auch hier schwierig zu entscheiden, insbesondere wenn eine klare Zielgruppen-Definition fehlt.

5. Fehlende Ressourcen

Aus dem bislang Beschriebenen wird deutlich, dass OSS-Projekte ein zentrales Problem haben: Es fehlen verfügbare Usability-Ressourcen, die zu einer verbesserten Benutzbarkeit für durchschnittliche Nutzer führen. Die Community besteht traditionell aus Programmierern, hingegen fehlen Usability-Experten fast vollständig. Selbst sehr große Desktop-Projekte, wie z. B. KDE, haben nur zwei bis drei Mitglieder, die eine profunde Kenntnis von Usability haben. Natürlich gibt es die Ausnahme, dass Firmen mit den eigenen Experten ein OSS-Projekt unterstützen (z. B. OpenOffice und Sun). Dahinter steckt jedoch ein kommerzielles Interesse, das nicht immer auf Gegenliebe stößt. Und der oben beschriebene Konflikt, ob man für die Community entwickelt oder für eine unbekannte Masse, wird dadurch eher verstärkt.

6. Geschwindigkeit als zentraler Vorteil

Bisher habe ich vor allem von den Schwierigkeiten für Open-Source-Projekte gesprochen, benutzerfreundliche Software zu produzieren. Es gibt jedoch einige Punkte, die OSS gerade gegenüber der klassischen Software-Entwicklung einen klaren Vorteil verschaffen. Der erste und vielleicht wichtigste Punkt ist das Prinzip „veröffentliche frühzeitig und häufig". Professionelles Usability-Engineering bezieht einen Teil seiner Stärke daraus, dass frühzeitig im Entwicklungsprozess Nutzer mit der Software konfrontiert werden können. Je früher diese Tests durchgeführt werden, desto leichter und kostengünstiger sind Veränderungen und Anpassungen. Deshalb werden Prototypen eingesetzt, die allerdings nicht immer so früh bereitstehen, wie es nötig wäre. Für OSS ist fortlaufende Veröffentlichung von inkrementellen Prototypen schlicht Teil des Prozesses – ein Traum-Zustand für jeden Usability-Experten! Durch die häufigen Releases wird es auch möglich, Verbesserungen nach und nach einzubauen. Klassische

Software muss sich hingegen auf große Releases konzentrieren, die keine Nachbesserungen mehr erlauben. Damit hat OSS einen strukturellen Vorteil, der potentiell zu besserer und einfacher zu bedienender Software führen kann. Diesen Vorteil gilt es zu nutzen, doch setzt er Usabilty-Ressourcen voraus.

7. OpenUsability

In der OSS-Entwicklung fehlt es bislang an Usability-Experten, und es fehlen erprobte Verfahren, diese einzubeziehen. Auf der Seite der potentiellen Usability-Kontributoren fehlt das Wissen, wie OSS-Entwicklung funktioniert, was getan werden kann und wie der Input gestaltet sein muss. Auf der Entwicklerseite ist unklar, was sie erwarten können, was gebraucht wird und ob den Vorschlägen der (häufig selbsternannten) Usability-Experten vertraut werden kann. Die Konsequenz ist beispielsweise, dass Anfragen in Mailinglisten oder per E-Mail unbeantwortet bleiben und die Sache im Sand verläuft.

Dies zu ändern ist eines der zentralen Anliegen des Projektes „OpenUsability". Die Idee dazu entstand auf der KDE-Entwickler-Konferenz 2003 in Nove Hrady, wo das Interesse an und die Bereitschaft zur Umsetzung von Usability sehr groß war, aber ein konkreter Ansatz, wie dies in der Praxis aussehen könnte, fehlte. Daraufhin wurde ein allen OSS-Projekten offenstehendes Portal konzipiert und entwickelt, das die Kommunikation und den Austausch zwischen Entwicklern und (externen) Usability-Experten ermöglicht.[4] Dazu gehören verschiedene Konzepte, die für beide Seiten zu einem annehmbaren Prozess führen sollen. So kann ein Projekt sich darin präsentieren und damit die Bereitschaft zeigen, Usability-Input umzusetzen. Umgekehrt können Usability-Experten direkt mit den verantwortlichen Projektmitgliedern in Kontakt treten und sich ein Bild über den Stand der Dinge verschaffen. Schließlich stehen zahlreiche *HOWTOs* zur Verfügung, die in die Besonderheiten der OSS-Entwicklung einführen und damit helfen, falsche Erwartungen zu vermeiden. Verschiedene Werkzeuge helfen, die Zusammenarbeit zu koordinieren und einen funktionierenden *Workflow* zu erreichen. Letzteres wurde insbesondere mit dem Projekt KDE-PIM (KDE Personal Information Management Tool) mehrfach durchgespielt und auf Praxistauglichkeit überprüft.

8. Neue Ressourcen

Gegenwärtig ist nicht eindeutig abzuschätzen, ob und in welchem Maße hier eine – entsprechend der Open-Source-Programmierung – freiwillige und unentgeltliche Unterstützung stattfinden wird. Jedoch hat „OpenUsability" bereits großes positives Feedback erhalten – und das, obwohl das Portal noch nicht fertiggestellt ist. Es scheint tatsächlich, als ob teilweise einfach die Schnittstelle gefehlt hat. Wenn „OpenUsability" diese Lücke füllen kann, wäre ein wichtiger Schritt getan. Es bleibt hoffentlich nicht

4 „OpenUsability" wurde von der relevantive AG als non-profit-Projekt konzipiert und entwickelt. Es ist quelloffen und wird voraussichtlich in Kürze in einen Verein übergehen http://www.openusability.org.

das einzige Projekt dieser Art. Unsere Hoffnung liegt darüber hinaus auf Universitäten und anderen Ausbildungseinrichtungen. So ist es vorstellbar, dass in den (viel zu wenigen) Kursen zu Ergonomie oder Usability anhand von Open-Source-Software gelehrt wird und Praxis-Projekte zur tatsächlichen Verbesserung von Software beitragen.

9. Die Zukunft

Open Source hat spezifische Nachteile bezüglich Benutzerfreundlichkeit. Dazu gehört insbesondere die traditionelle Orientierung an einer sehr Computer-affinen Nutzergruppe und damit der Vernachlässigung von „normalen Nutzern". Auch die starke Bevorzugung von Funktionalität gegenüber Zugänglichkeit und Benutzbarkeit gehört dazu. Damit einher geht die bisher schwache Einbindung von Usability-Experten und als Konsequenz die geringe Verfügbarkeit von Usability-Ressourcen. Es gibt jedoch starke Anzeichen dafür, dass sich dies ändert bzw. ändern kann. Das Interesse im KDE-Projekt an Usability und die Schritte, die zur Verbesserung der Software unternommen werden, sind sehr ermutigend. Nimmt Usability allgemein einen stärkeren Platz in der OSS-Entwicklung ein, kann es seine Stärken, insbesondere die schnelle, im Ansatz prototypische Entwicklung, voll ausspielen.

Unter diesen Voraussetzungen halte ich es für möglich, dass OSS mittelfristig gerade durch Benutzerfreundlichkeit seinen Siegeszug auf dem Desktop begründet.

Literatur

Gruber, J. (2004), 'Ronco-Spray on Usability', http://daringfireball.net/2004/04/spray_on_usability.

Mühlig, J., Horstmann, J., Brucherseifer, E. und Ackermann, R. (2003), Linux Usability Studie, techn. Bericht, relevantive AG. http://www.relevantive.de/Linux-Usabilitystudie.html.

Nichols, D. M. und Twidale, M. B. (2002), Usability and Open Source Software. Working Paper 02/10, Department of Computer Science, University of Waikato, New Zealand. http://www.cs.waikato.ac.nz/~daven/docs/oss-wp.html.

Nielsen, J. (2004), 'Developer Spotlight: Jakob Nielsen', Builder AU http://www.builderau.com.au/webdev/0,39024680,39130602,00.htm.

Raymond, E. S. (1998), 'The Cathedral and the Bazaar', *First Monday* **3**(3). http://firstmonday.org/issues/issue3_3/raymond/.

Raymond, E. S. (2004), 'The Luxury of Ignorance: An Open-Source Horror Story', http://www.catb.org/~esr/writings/cups-horror.html.

Der Beitrag freier Software zur Software-Evolution

ANDREAS BAUER UND MARKUS PIZKA

Ohne ein gesondertes Interesse an der Thematik Software-Evolution zu haben, hat die stetig wachsende Open-Source-Bewegung über die letzten zwei Jahrzehnte trotzdem auf bemerkenswerte Weise äußerst erfolgreiche Konzepte, Methoden und Techniken für die kontinuierliche und kontrollierte Weiterentwicklung komplexer Software-Systeme etabliert. Diese Evolutionstechniken repräsentieren *Best Practices*, die ebenso zur Lösung aktueller und höchst kritischer Probleme bei der Pflege und Weiterentwicklung kommerzieller Software-Systeme herangezogen werden könnten. Im vorliegenden Artikel stellen die Autoren einige dieser Prinzipien aus der Perspektive erfahrener Open-Source-Entwickler dar. Dabei wird deutlich, dass der extrem dynamische Prozess der Entwicklung freier Software eng mit dem konstanten Wachstum der Code-Basen und stets veränderlichen Projektgrößen verbunden ist. Weiter wird argumentiert, dass ein Großteil des Erfolgs von Open-Source-Software tatsächlich auf den erfolgreichen Umgang mit diesem kontinuierlichen Wandel zurückzuführen ist.

1. Einführung

> „Virtue is more to be feared than vice, because its excesses are not subject to the regulation of conscience." – Adam Smith (1723–1790)

Die initiale Entwicklung und stetige Evolution von Open-Source-Software-Produkten weist erstaunliche Ähnlichkeiten mit den Ideen der freien Marktwirtschaft auf. Die liberale Bereitstellung, Nutzung und Veränderung von Software, wie sie in Open–Source-Umgebungen praktiziert werden, teilen die Prinzipien des Wirtschaftsliberalismus (Smith 1776), indem sie einen Raum für ungehinderten Handel, Veränderung und Wachstum etablieren. Infolgedessen könnte man spekulieren, dass sich die Entwicklung von Produkten und Prozessen in Open-Source-Umgebungen auf lange Sicht gesehen als überlegen gegenüber allen anderen Modellen herausstellen wird, weil das selbstregulierende Wechselspiel zwischen Angebot und Nachfrage für kontinuierliche Selektion und Verbesserung sorgen wird.

Tatsächlich gibt es zunehmend Hinweise für diese Hypothese. Die anfangs eher unbedeutende Open-Source-Gemeinde hat sich zu einem umfangreichen freien Markt

für Software-Artefakte mit tausenden Teilnehmern und unzähligen Produkten weltweit entwickelt. Aufgrund der enormen Energie in diesem Markt haben einige Produkte, wie GNU/Linux oder der Übersetzer GCC eine hinreichend hohe Qualität für breiten kommerziellen Einsatz erreicht. Zusätzlich kann dieser Markt aufgrund seiner Größe und Vielfalt schnell auf verändernde bzw. neue Anforderungen reagieren – z. B. die Entwicklung neuer Gerätetreiber. Neben den Produkten entwickelt sich auch der Entwicklungsprozess selbst innerhalb dieses Marktes ständig weiter. Die Entstehung strukturierter Kommunikationsplattformen, [1] die Einführung von Rollen – *Maintainer* – und die Verbreitung von Elementen agiler Methoden (Beck 1999) stützen diese Behauptung.

Nun soll dieser Artikel nicht den Eindruck erwecken, Open Source sei die *silver bullet* (Brooks Jr. 1995) des Software-Engineerings, denn es ist offensichtlich, dass auch dieses Modell inhärente Defizite besitzt und nicht in jeder Situation zwangsläufig zu besseren Ergebnissen führen muss. Demgegenüber wird im Folgenden argumentiert, dass im Zuge freier Software in der Vergangenheit Konzepte entstanden sind, die potentiell auch großen kommerziellen Softwareprojekten, mit Millionen von Zeilen von Code, dabei helfen können, sich sogar über Jahrzehnte auf „gesunde" Art und Weise weiterzuentwickeln. Unsere eigenen praktischen Erfahrungen mit einigen weithin bekannten Open-Source-Produkten wie der GNU Compiler Collection (GCC) oder GNU/Linux sowie unsere Urheberschaft und Beteiligung an kleineren Open–Source-Produkten, haben uns den Nutzen des Entwicklungsmodells freier Software sowie den daraus resultierenden Architekturen klar vor Augen geführt.

Anhand dieser Erfahrungen können wir guten Gewissens argumentieren, dass einige der nachstehend diskutierten Open-Source-Prinzipien und -Praktiken in kommerzielle bzw. proprietäre Software-Organisationen zur kontinuierlichen Verbesserung bestehender Produkte und Prozesse übernommen werden können und sollten.

Abschnitt 2. klärt den Kontext unserer Arbeit und grenzt die Gültigkeit der Aussagen ein. Dabei wird der Begriff Open Source als „freie Software" präzisiert, es werden verwandte Arbeiten zur Evolution freier Software beschrieben und ein Zusammenhang mit Forschungsarbeiten im Bereich Software-Evolution hergestellt. Danach werden wir in 3. einen detaillierten Blick auf die Evolution der Prozesse in freien Softwareprojekten werfen, bevor wir uns in 4. auf die damit zusammenhängende Evolution der technischen Architektur der Produkte konzentrieren werden. In 5. schließen wir den Artikel mit einer kurzen Zusammenfassung der wesentlichen Resultate unserer Studie ab und diskutieren die Übertragbarkeit der Ergebnisse auf nichtfreie Software-Systeme und -Umgebungen.

2. Open Source und Freie Software

Wenn wir uns in diesem Artikel auf Open-Source-Software beziehen, beschränken wir uns nicht auf die aktuell sehr populäre Open-Source-Bewegung an sich, die größ-

1 Siehe SourceForge, Open Source Technology Group: http://sourceforge.net/.

tenteils durch das GNU/Linux-Projekt[2] inspiriert wurde. Wir sprechen vielmehr von Software, deren Nutzungs- und Änderungsrechte dem (sogar noch älteren) Ideal der freien Software an sich entsprechen. Diese Freiheit beschränkt sich nicht auf eine Erlaubnis zum Lesen des Quellcodes, wie bei einer „Redefreiheit", sondern erlaubt uneingeschränkte und kostenlose Nutzung, Weitergabe, Modifikation, ganz wie beim allseits beliebten „Freibier". Der Hauptunterschied liegt also darin, dass Redefreiheit in der zivilisierten Welt eher als ein menschliches Grundrecht anzusehen ist, mit dem noch garantiert ist, dass damit auch etwas erreicht werden kann, wohingegen Freibier einzig und alleine deshalb positiv auffällt, weil es nichts kostet und die künftige Verwertung gänzlich offen lässt.

Gemäß der GNU GPL[3] der Free Software Foundation (FSF) – die Lizenz des GNU/Linux-Projekts – darf Quellcode modifiziert und von einer dritten Person verwendet werden, solange diese Programmmodifikationen ebenso frei und offen bleiben und die ursprünglichen Copyright-Hinweise nicht verändert werden. Die FSF glaubt, dass dies der beste Weg sei, das Recht der Nutzer zum Verstehen und ggf. Anpassen von Programmen zu erfüllen. Wenngleich diese Freiheiten schon sehr weit gehen, sind BSD-artige Lizenzen[4] noch viel liberaler und erlauben einer dritten Person mehr oder weniger mit den Sourcen zu machen, was sie will – sogar Modifikationen der freien Software als proprietäre *Closed Source* zu vertreiben.

Der Rest dieses Artikels beschäftigt sich also mit den Erfahrungen der Autoren im Umgang und in der Entwicklung von *freier* Software im Allgemeinen, anstatt mit ihren jeweiligen speziellen Ausprägungen, die unter anderem auch Trends obligen. Die Begriffe Open Source und „freie Software" werden synonym für Software verwendet, die ähnlich wenigen Nutzungsbeschränkungen unterliegt wie Freibier.

2.1. Mythen und Klischees

Eine weit verbreitete falsche Vorstellung ist, dass Open Source oder *free software* in einem chaotischen Prozess von Freizeit-Hackern, die nicht die gleichen Ansichten bezüglich Wartbarkeit haben wie ihre „professionellen" oder akademischen Entsprechungen, erstellt wird und in mehr oder weniger brauchbaren Software-Produkten mündet. Bedauerlicherweise kann auch der Titel des bekannten *Papers* von Eric Raymond „The Cathedral and the Bazaar" (Raymond 1998) leicht dahingehend missverstanden werden, wodurch das Chaos-Klischee zusätzlich unterstützt wird.

Doch selbstverständlich ist das weit gefehlt: Freie Software entsteht nicht auf einem chaotischen Basar. Wie wir unten zeigen werden, ist die Entwicklung freier Software oft sehr gut organisiert und bedient sich strukturierter Prozesse mit klar definierten Rollen. Infolgedessen gibt es eine große Anzahl freier Software-Produkte, deren Qualität kommerziellen Varianten mindestens ebenbürtig ist und folglich unsere These unterstützt.

2 GNU/Linux: http://www.linux.org/
3 GNU General Public License: http://www.fsf.org/licenses/gpl.html
4 The BSD License: http://www.opensource.org/licenses/bsd-license.html

Leider muss sich die Freie-Software-Bewegung allerdings sehr wohl vorwerfen lassen, dass sie die Resultate und Trends in der Forschung im Bereich der Software-Evolution ignoriert. Umgekehrt ignoriert die andere, nicht-freie Seite die Umstände, unter denen ihre Lieblingstexteditoren, Compiler oder Betriebssysteme ins Leben gerufen wurden. Dieser Artikel zielt darauf ab, diese Lücke zwischen den Errungenschaften der freien und der „professionellen" Softwareentwickler aus dem akademischen Umfeld und in der Industrie ein Stück weit zu schließen.

2.2. Wirklichkeit

Um nur einige wenige der erfolgreichen freien Software-Systeme zu nennen, weisen wir auf die BSD-basierten Betriebssysteme FreeBSD, NetBSD, OpenBSD und Darwin[5] hin, die ihren Ursprung hauptsächlich in Ideen und Code aus den 80er Jahren haben. Dank der hohen Qualität dieser Produkte, die sich über einen langen Zeitraum entwickelt hat und der „Freibier"-Philosophie der BSD-Lizenz, basieren kommerzielle Betriebssysteme von wichtigen Anbietern wie Apple heute auf einem BSD *Open Source Kernel*, der nach Charles Darwin – dem Urvater der Evolutionstheorie (Darwin 1859) – benannt ist. Dies unterstreicht die Bedeutung der erfolgreichen Evolution von Software-Produkten, da insbesondere bei solchen komplexen Systemen eine langfristige erfolgreiche Reifung notwendig ist, um die notwendige Qualität zu erreichen.

Sogar Free-speech-Projekte wie GCC oder GNU/Linux werden heute zu einem großen Teil von dem finanziellen Engagement großer Konzerne wie IBM oder Red Hat getragen, die versierte Entwickler, Geld und andere Ressourcen für Softwareprojekte zur Verfügung stellen, bei denen jeder den Quellcode lesen und verändern kann (Harris et al. 2002). Das Ergebnis muss natürlich, ganz entgegen der verbreiteten Fehleinschätzung des Chaos-Prozesses, ein wartbares Produkt sein, da Wartbarkeit ein wesentlicher Grund dafür ist, dass so alte „Dinosaurier-Projekte" wie GCC, *BSD, Emacs, GNU/Linux usw. heute in ihren Bereichen nach wie vor erfolgreich sind.

Offensichtlich hat die Open-Source-Bewegung ihre eigenen Konzepte und Techniken entwickelt, die es großen Softwareprojekten erlaubt, sich auch bei ständig wechselnden Anforderungen langfristig erfolgreich und gesund weiterzuentwickeln. Obwohl viele der freien Software-Produkte unter dem Aspekt der technischen Innovation nicht mehr interessant sind, sind sie heutzutage alles andere als irrelevant. Ein Teil des offenkundigen Erfolges eines Betriebssystems wie GNU/Linux muss deshalb zwangsläufig an der Art und Weise liegen, wie sich diese Software an Veränderungen sowohl der technischen Realität als auch der Anzahl der Personen, die etwas zu dem Projekt beitragen (siehe Abschnitte 3.1., 3.2.), anpasst.

2.3. Verwandte Arbeiten zur Software-Evolution

Aufgrund der großen Lücke zwischen proprietärer Softwareentwicklung, akademischer Forschung und den Praktiken der freien Software-Gemeinde ist es nicht ver-

5 (Open)Darwin http://www.opendarwin.org/, http://developer.apple.com/darwin/

wunderlich, dass nach wie vor nur wenige Forschungsarbeiten auf die Weiterentwicklung freier Software-Produkte abzielen. Ausnahmen sind Nakakoji et al. (2002), Lehey (2002), Succi und Eberlein (2001) und Godfrey und Tu (2000). Auf der anderen Seite spielen in der Welt der freien Software eine beträchtliche Anzahl praktischer Konzepte und Techniken für die Evolution vorhandener Software eine wichtige Rolle, wie zum Beispiel Konfigurationsmanagement (z. B. CVS), Regressionstests (Savoye 2002), *Refactoring* (Opdyke 1992), Analyse von Quell-Code, Code-Generatoren und *Separation of Concerns* – um nur einige zu nennen. Wir verzichten bewusst auf die weitere Aufzählung dieser langen Liste, aber es sollte offensichtlich werden, dass die kontinuierliche Weiterentwicklung freier Software, wie sie in dieser Arbeit beschrieben wird, zahlreiche Verbindungen mit verschiedenen konkreten Evolutionstechniken besitzt. Indes konzentrieren wir uns in dieser Arbeit auf die *Prinzipien* der Evolution freier Software, unabhängig von bestimmten Techniken.

Lehmans wohlbekannte acht Gesetze der Software-Evolution (von Lehman 1969 bis Lehman und Ramil 2001) zielen auf die fundamentalen Mechanismen ab, die der Dynamik der Software-Evolution zugrunde liegen. Wie wir sehen werden, entspricht freie Software den Gesetzen Lehmans ganz hervorragend und das, obwohl Lehmans Gesetze und der Entwicklungsprozess freier Software unabhängig voneinander entstanden sind: Freie Software verändert sich ständig (Gesetz I), die Komplexität nimmt zusehends zu (Gesetz II) und die Selbstregulierung des Evolutionsprozesses ist offensichtlich (Gesetz III). Unsere im Folgenden beschriebenen Beobachtungen stützen auch die verbleibenden Gesetze IV bis VIII. Im Gegenzug ist die weitreichende Übereinstimmung der freien Software-Entwicklung mit Lehmans Gesetzen eine Möglichkeit, den Erfolg der Open-Source-Softwareentwicklung zu erklären.

3. Evolution des Entwicklungsprozesses

Anders als bei vielen proprietären Softwareprojekten beginnt die Entwicklung freier Software meist ohne jeden zusätzlichen Verwaltungsaufwand. Typische frühe Projektphasen zu Strukturierung und Koordination der noch folgenden Phasen, wie zum Beispiel eine exakte Anforderungsanalyse, spielen oftmals überhaupt keine Rolle. Ebenso ist es jedoch offensichtlich, dass der administrative Aufwand nicht konstant bleiben kann, wenn das Projekt wächst. Hierfür gibt es viele bedeutende Beispiele (Lehey 2002, Nakakoji et al. 2002, Cubranic und Booth 1999).

In der Tat ist der Entwicklungsprozess bei freier Software höchst dynamisch, skaliert mit der darunterliegenden Architektur und auch mit der Anzahl und den Fähigkeiten der Menschen, die am Projekt beteiligt sind. Den *einen* Entwicklungsprozess für freie Software gibt es allerdings nicht, sondern sich weiter entwickelnde Prozesse, die stark mit der Komplexität der resultierenden Produkte selbst verwoben sind.

3.1. Unausweichliche technische Veränderungen

Einige der Veränderungen im Entwicklungsprozess freier Software beruhen eher auf technischen Veränderungen als auf Entscheidungen der Entwickler. Zum Beispiel

war der weit verbreitete Übersetzer GCC ursprünglich Mitte der 80er Jahre als schneller und praxisnaher C-Compiler auf 32-Bit-Plattformen entwickelt worden, welche 8-Bit-Bytes adressieren und mehrere Allzweckregister[6] besitzen. Heutzutage unterstützt der GCC mehr als 200 verschiedene Plattformen (Pizka 1997) sowie zahlreiche weitere Programmiersprachen. Sein Kern besteht aus über 900 000 Programmzeilen, und während das GCC Projekt aus einem zunächst simplen E-Mail-Verkehr (später auch über Usenet) zwischen Entwicklern des Kern-Teams entstanden ist, funktioniert der derzeitige Entwicklungsprozess heute grundlegend anders. Das Projekt hat offensichtlich nicht nur seine ursprünglichen Ziele geändert, sondern auch die Anzahl und auf gewisse Weise auch die Art der Mitwirkenden in Bezug darauf, wie sie sich an der fortlaufenden Evolution von GCC beteiligen. Das Handbuch der Version 3.2.2 [7] listet die Namen von 302 verschiedenen Mitwirkenden auf, was natürlich ein starker Kontrast zu 1984 ist, als Richard Stallman die erste Version eines damals eher einfachen System-Compilers für potenzielle GNU-Plattformen alleine erstellte.

Diese drastischen Veränderungen wurden hauptsächlich durch die technischen Fortschritte möglich, die sich seit der Geburt von GCC ereignet haben. Hierzu gehören besonders die weltweite Ausbreitung des Internets mit all seinen neuen Transfer–Protokollen, die während des Projekts entstanden sind (siehe das *hypertext transfer protocol*, http). Im Speziellen zieht das GCC-Projekt heute aus den folgenden technischen Fortschritten seine Vorteile:

- Automatisches Management von Mailinglisten mit Zugriff auf durchsuchbare Archive und Web-Oberflächen: so werden die Bemühungen eines weltweit verbreiteten Netzwerks von Entwicklern koordiniert und gesteuert,
- (Öffentliche) CVS-Server mit Web-Oberflächen, die sowohl verschiedene Versionen als auch unabhängige Entwicklungen innerhalb des Projekts verfolgen,
- Eine große Anzahl von (http und ftp) *mirror sites*, welche die Verfügbarkeit der relevanten Daten weltweit erhöhen,
- Die Einführung und das Interesse an neuen Sprachen (z. B. Java, Haskell) und neuen Hardware-Plattformen (z. B. IA-64-Architektur),
- Ein modernes und automatisiertes Fehlerverfolgungssystem, das weltweit via World Wide Web zugänglich ist,
- *compile farms*, die an einem Punkt zentralen Zugang zu mehreren Plattformen bieten und für gewöhnlich durch Firmen aus der Industrie gefördert werden, die ein beträchtliches Interesse an Open-Source-Produkten haben,
- Eine wachsende Anzahl peripherer Projekte, die auf dem GCC aufbauen, aber ihren eigenen Fortschritt unabhängig managen (z. B. Glasgow Haskell Compiler (GHC), Echtzeit-Java-Implementierung, Mercury Compiler).

6 GNU Compiler Collection Internals: http://gcc.gnu.org/onlinedocs/gccint/

7 Using the GNU Compiler Collection: http://gcc.gnu.org/onlinedocs/gcc-3.2.2/gcc/

Projekt	Alter	Code-Zeilen
Linux kernel 2.4.2	1991	2 437 470
Mozilla	1998	2 065 224
XFree86 4.0.3	1987	1 837 608
GCC 2.96-20000731	1984	984 076
GDB / DejaGnu-20010316	Mitte 1980er	967 263
Binutils 2.10.91.0.2	Mitte 1980er	690 983
glibc 2.2.2	Frühe 1990er	646 692
Emacs 20.7	1984	627 626

Tabelle 1: Größe von Open-Source-Projekten (Wheeler 2002)

Das gilt auch für andere große und erfolgreiche Open-Source-Projekte, die sich meistens durch Anwendung derselben Tools (CVS, RCS, BugZilla usw.) unter ähnlichen Umständen oder technischen Bedingungen weiterentwickeln. Weitere Beispiele sind *BSD, Mozilla und der Linux Betriebssystem-Kernel. Wenn wir die wachsende Anzahl der Mitwirkenden, z. B. bei FreeBSD und GCC vergleichen, sehen wir Ähnlichkeiten zwischen beiden Projekten: Im Jahre 1995, als FreeBSD 2.0 freigegeben wurde, hatte es 55 Mitwirkende (d. h. Anwender mit Schreibzugriff auf das *code repository*); bis Ende 2002 war es insgesamt 319 Leuten gestattet, Änderungen selbst direkt an der Code-Basis einzupflegen (Lehey 2002).

3.2. Änderung der Organisation

Die wahre Herausforderung beim Aufbau und Unterhalt eines erfolgreichen Entwicklungsprozesses für ein freies Software-Produkt besteht darin, technische Veränderungen entsprechend den begleitenden sozialen Veränderungen einzuführen und zwar dann, wenn das Projekt eine „kritische Größe" überschreitet. Die kritische Größe hängt nicht allein von der Anzahl der Zeilen im Quell-Code ab, sondern auch von den Programmmodulen, der Anzahl der Mitwirkenden und im Grunde genommen auch von beliebigen Metriken, die es uns erlauben, Schlussfolgerungen über die Komplexität eines Software-Produktes zu ziehen.

Die heutzutage florierenden freien Softwareprojekte haben alle ihre kritische Größe mindestens einmal überschritten, gewöhnlich in der Anzahl der Zeilen und oft auch im Hinblick auf die Zahl der Mitwirkenden. Wie aus Tabelle 1 ersehen werden kann, besteht die Linux-Version 2.4 aus mehr als 2 400 000 Code-Zeilen und außerdem deuten ihre derzeitigen *changelogs* eine mindestens gleich große Zahl von aktiven Entwicklern an, wie sie etwa an FreeBSD beteiligt sind.

Diese Zahlen weisen auf die Tatsache hin, dass organisatorische Aufgaben bei erfolgreicher freier Software nicht durch eine einzelne Person gelöst werden können (z. B. durch den ursprünglichen Initiator eines Projekts). Wenn eine kritische Größe einmal erreicht ist, entsteht so oder so eine „Geschäftsführung", und das Gremium

Projekt	Name des Gremiums	Mitglieder
FreeBSD	Core	15
GCC	Steering Committee	12
Debian	Leader & Technical Committee	8
Mozilla	Project Managers („Drivers“)	13
KDE	Core Group	20

Tabelle 2: Führungsgremien in freien Softwareprojekten

koordiniert und kontrolliert die weitere Evolution des Produktes. In Konsequenz daraus wird ein reifes Projekt wie GCC nicht mehr von Richard Stallmann vorangetrieben, sondern von einem „Lenkungsausschuss“, der heute aus 12 professionellen Software-Entwicklern besteht, die teilweise von Software-Anbietern bezahlt werden (z. B. Red Hat, IBM, Apple), um sich auf die Entwicklung einer freien *compiler suite* zu konzentrieren (siehe Tabelle 2). Während die Mitglieder alle Experten auf ihren Gebieten sind, wird von keinem einzelnen erwartet, ein vollständiges Verständnis für den gesamten Code von 900 000 Zeilen zu haben.

Wie aus Tabelle 2 ersehen werden kann, scheinen freie Software-Produkte gut mit einem Lenkungsausschuss bestehend aus 10–20 Mitgliedern auszukommen. Offensichtlich ist diese Mitgliederzahl groß genug, um ein komplexes Projekt zu managen, aber klein genug, um sicherzustellen, dass die Entwicklung nicht aufgrund interner Debatten und unglücklicher Politik gehemmt wird. Interessant ist auch, dass sich die aufgelisteten Projekte bezüglich Größe und Alter sehr ähnlich, aber historisch kaum miteinander verbunden sind und dennoch gemeinsam 10–20 Kernmitglieder als eine angenehme Größe empfinden, um die „richtigen“ Entscheidungen für das jeweilige Projekt zu treffen.

Es sollte an dieser Stelle herausgestellt werden, dass die Strategie auch perfekt zu den Ergebnissen umfangreicher Studien über den Erfolg und das Scheitern von kommerziellen Softwareprojekten passt, wie z. B. die wohl bekannten CHAOS-Berichte der Standish Group (The Standish Group International 1999, 1995). Diese Studien zeigen eine Tendenz zum Scheitern eines Projektes, je größer es wird. Daher wird stark empfohlen, ein Projekt überschaubar klein zu halten, da es ansonsten sehr wahrscheinlich scheitern wird. Das ist genau der Grund, warum agile Methoden wie Extreme Programming (XP) sich absichtlich weigern, „riesige“ Projekte in einem einzelnen Schritt auszuführen. Unglücklicherweise scheint die Verbreitung dieser grundlegenden Lektion in kommerziellen Umgebungen sehr langsam zu sein. Im Gegensatz dazu gewährleistet die Selbstregulierung durch Wettbewerb und Auslese innerhalb freier Softwareprojekte eine passende Projektgröße. Das Projekt wird je nach Bedarf fortlaufend in kleinere Teilbereiche aufgespalten, wobei den Mitgliedern des Entwicklerteams neue Wartungsaufgaben zugeordnet werden, die ihrem Können, ihrer Erfahrung und ihren aktuellen Kompetenzen entsprechen. Letztendlich skaliert die gesamte Organisation mit der Gesamtgröße des Produktes.

Der Weg, wie der Führungsausschuss nominiert wird unterscheidet sich bei den verschiedenen freien Softwareprojekten. Während Debian zum Beispiel demokratische Wahlen einsetzt, beruht die Auswahl im KDE-Team ausschließlich auf den Verdiensten der Mitglieder um das Projekt. Jeder kann Mitglied der KDE-Kerngruppe werden, aber der spezielle Bewerber muss sich selbst durch herausragende Beiträge und Hingabe über einen beachtlichen Zeitraum hinweg ausgezeichnet haben. Dessen ungeachtet stellen sowohl KDE als auch Debian sicher, dass vorrangig die passendsten und begabtesten Leute die Verantwortung übernehmen und nicht die, die am lautesten schreien, am ältesten oder jüngsten sind usw. – wie es oft in kommerziellen Umgebungen der Fall ist. Dies ist für sich ein Evolutionsprozess, der Personen mit Wissen und praktischen Fähigkeiten fördert anstatt reine Autoritätspersonen.

3.3. Konfigurations- und Änderungsmanagement

Trotz der Ähnlichkeiten zwischen den verschiedenen „großen" freien Softwareprojekten, gibt es viele verschiedene Ansätze für das Änderungsmanagement. Diese konvergieren jedoch mehr und mehr aufgrund des Einsatzes ähnlicher Werkzeuge (CVS, RCS, BitKeeper, patch/diff usw.), welche die teilweise sehr komplexen Aufgaben automatisieren.

Wie Nakakoji et al. (2002) beobachtet haben, scheinen die verschiedenen Ansätze im Open-Source-Änderungsmanagement in etwa gleich erfolgreich zu sein, da es bedeutende Beispiele für den Erfolg der einzelnen Techniken gibt. In ihrem Dokument werden die verschiedenen Änderungshistorien „Evolutionsmuster" genannt, welche sich selbst hauptsächlich durch die Art abgrenzen, wie Änderungen getestet, geprüft und mit dem Projekt zusammengeführt werden.

Linus Torvalds und Alan Cox, zwei treibende Kräfte der Linux-Kernel-Entwicklung, haben in einigen Interviews Bedenken gegenüber dem Gebrauch von öffentlichen CVS-Servern zur Abwicklung des Konfigurationsmanagements erhoben, und das, obwohl gleichzeitig viele wichtige Linux-Module auf diese Weise unter Versionskontrolle stehen (Southern und Murphy 2002). Auf der anderen Seite scheinen GCC und *BSD mit den Einrichtungen und den Möglichkeiten öffentlicher CVS-Server zufrieden zu sein, trotz gelegentlicher *commit wars* (Lehey 2002). Von einem *commit war* wird dann gesprochen, wenn Entwickler, die an denselben Teilen des Codes arbeiten, laufend die Änderungen des jeweils anderen überschreiben. Bei dem wichtigsten Linux-Kernel (dem von Linus Torvalds) kann das nicht passieren, weil Linus Torvalds als der Projektleiter persönlich entscheidet, welche Änderungen angenommen und welche lieber fallen gelassen oder durch andere *Maintainer* für ihre eigenen Entwicklungsbäume aufgegriffen werden.

Bezüglich des Änderungsmanagements scheinen Open-Source-Projekte dieselben Dinge zu tun, wie sie bei proprietärer Software vorgefunden werden können (siehe auch van der Hoek 2000). Man sollte auch darauf hinweisen, dass sogar CVS selbst seine Wurzeln in der freien Software-Welt (Cederqvist et al. 1993) hat, was einfach zeigt, wie sich ein Open-Source-Programm, das als eine Hand voll Shell-Skripten, die im Usenet verbreitet wurden, begann, sich zu einem De-facto-Standard entwickelt

hat – eben weil es stets fest integriert in die Entwicklung anderer freier Software–Produkte war und deren Unterstützung zum Ziel hatte. Im Klartext heißt das, dass die zahlreichen Produkte, die mit Hilfe von CVS entwickelt wurden, tatsächlich den Entwicklungsprozess dieses *toolkits* selbst ausgelöst haben, bis hin zu dem Punkt, an dem es ein unersetzlicher Eckpfeiler sowohl für freie als auch kommerzielle Projekte wurde.

4. Evolution der Architektur

In einem freien Software-Projekt ist es nicht nur die soziale Struktur, die sich einer sich verändernden Realität anpasst, sondern auch die Architektur selbst, die sehr oft eher einer natürlichen Entwicklung unterliegt, als das Ergebnis einer sorgfältig geplanten Anforderungsanalyse zu sein. Wiederum ein gutes Beispiel für diese Behauptung ist das kürzlich durch das GCC-Projekt angenommene, von Intel vorgeschlagene *cross vendor application binary interface*, das anfangs auf negative Resonanz der Nutzer stieß, weil es Inkompatibilitäten mit früheren C++ Programmen verursachte. Der Schritt war jedoch notwendig, um mit den Binärpaketen kompatibel zu sein, die von anderen (kommerziellen) Compilern erzeugt werden, die auch Intels neue (64-Bit-) Hardware und einen Großteil der von dem neuesten ISO-Standard für die Sprache C++ vorgeschlagenen Features unterstützen (Intel Corporation 2001).

4.1. Modulare Programme und Schichten

Das Verändern der ABI eines existierenden Compilers ist keinesfalls eine triviale Aufgabe. Im Falle des GCC allerdings war diese Änderung durch das modulare Design der Architektur möglich: Der Aufbau erlaubt es, den Großteil der Code-Generierung auf eine plattformunabhängige Art und Weise (siehe Abbildung 1) zu erledigen, denn viele der notwendig gewordenen Veränderungen im GCC Back-End sind für die späten Phasen der Übersetzung, in denen die Repräsentation der Register Transfer Language (RTL) auf plattformabhängige *templates* in Maschinensprache abgebildet wird, völlig transparent.

Andere Projekte wie der Linux-Kernel, Mozilla oder *BSD haben ähnliche logische und physische Programmmodule, die eine optimierte Koordination der Beiträge der Mitwirkenden und einen besseren Ansatz für das Änderungsmanagement ermöglichen. Die Mehrheit dieser Module aber war nicht notwendigerweise ersichtlich und hat folglich auch nicht bestanden, als diese Projekte vor mehr als zehn Jahren initiiert wurden. Deswegen muss diese vernünftige Strukturierung aus dem Evolutionsprozess selbst resultieren, der stattfindet, wenn eine wachsende Anzahl an Mitwirkenden neue Quelltexte, Ideen und Lösungen beisteuert, um den bereits existierenden Code an vielen Stellen zu verbessern.

4.2. Verstrickung von Prozess und Architektur

Wir behaupten also, dass in gut gewarteten, erfolgreichen freien Software-Produkten die technische Struktur der zugrunde liegenden Architektur stets mit der Organisa-

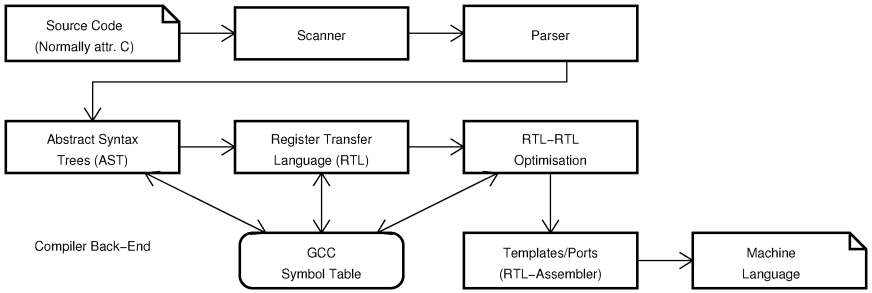


Abbildung 1: Logische und physische Module des GCC Kerns

tion des Projekts verstrickt ist. In anderen Worten: Jede wichtige Änderung in der Organisation des Projekts muss zwangsläufig zu einer Veränderung der technischen Architektur des Produkts führen und umgekehrt.

Abbildung 2 zeigt die starken Wechselbeziehungen zwischen der technischen Struktur des GCC und der Gesamtorganisation des Projekts. Sie zeigt auch, wie Ergänzungen zur Code-Basis zuerst durch eine öffentliche Überprüfung in Mailinglisten koordiniert werden, um offensichtliche Fehler zu eliminieren bevor sie in das *cvs repository* geschrieben werden. Im Falle des GCC durchlaufen sogar die *Maintainer* selbst diesen Prozess, um die anderen Mitwirkenden über die anstehenden Änderungen zu informieren. Tatsächlich ist es nicht ungewöhnlich, dass sich eine Änderung im Form eines Programmfragments (*patch*) noch während dieses Prozesses weiterentwickelt, bevor der *patch* ein fester Bestandteil der GCC-Suite ist. Der Grund dafür ist, dass große oder komplexe *patchs* gut verstanden werden müssen, bevor sie angenommen werden können. Folglich ist ein Prozess der konstanten Verfeinerung von Nöten, mit dem sich der *patch* an die sich kontinuierlich ändernde Code-Basis annähert. Unsere eigenen Erfahrungen in der GCC-Entwicklung haben gezeigt, dass dieser Prozess manchmal Wochen oder sogar Monate in Anspruch nimmt (Bauer 2003).

Folglich unterliegen die Gesamtorganisation und Reorganisation der Arbeiten der Mitwirkenden einem „natürlichen" Prozess, der hauptsächlich durch Notwendigkeiten und Begründungen getrieben wird und nicht durch Autorität. Aufgrund der starken Verstrickung von Entscheidungen bzgl. der Architektur und der Organisation des Projekts muss sich ein großer Teil der technischen Struktur genauso „natürlich" entwickeln: Die Lösung, die die anwendbarste, widerstandsfähigste oder die am leichtesten zu wartende ist, hat letztendlich Erfolg – vielleicht nicht sofort, aber sicherlich asymptotisch gesehen. Also eine „gesunde" Art der Evolution, die auch für viele proprietäre Projekte wünschenswert wäre, aber durch den Einfluss zusätzlicher Interessen nicht stattfindet.

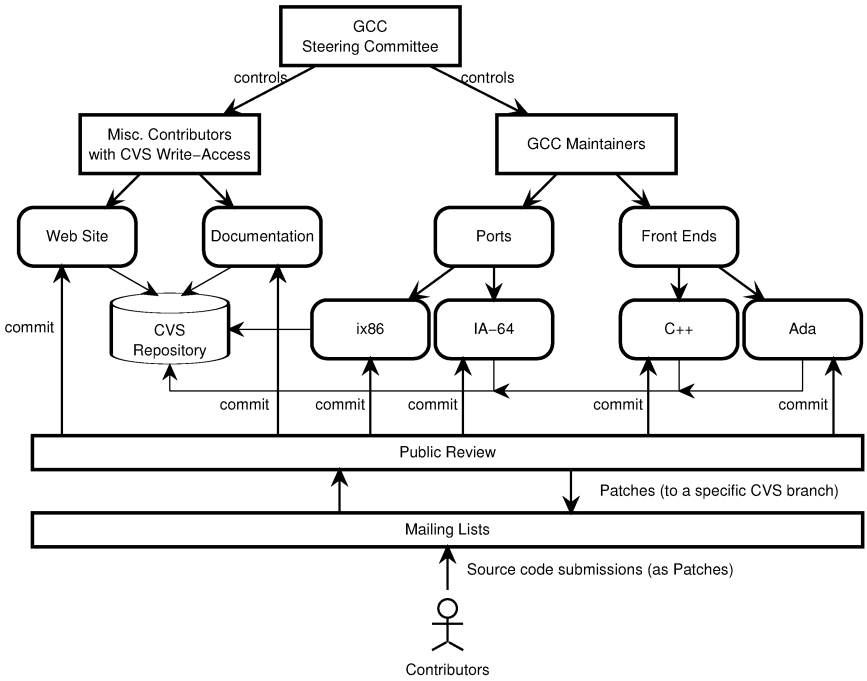


Abbildung 2: Die wesentlichen Elemente der GCC Projektstruktur und Organisation

4.3. Projekt- und Interprojekt-Abhängigkeiten

Obwohl kommerzielle Software-Anbieter natürlich um die Weiterentwicklung ihrer Software besorgt sind, haben sie oft eine statische, produktorientierte Projektorganisation, welche im strengen Kontrast zur Bedeutung des Wortes „Evolution" steht. In der Tat gibt es Fälle, in denen die Projektstruktur in proprietären Projekten mehr die geographische Verteilung der Firma widerspiegelt, als den Zweck und die Ziele des Produkts. Zum Beispiel ist das Team in Stadt *A* mit einer Aufgabe *a* betraut und das Team in Stadt *B* ist verantwortlich für Aufgabe *b*. Kann von Software wirklich erwartet werden, dass sie sich unter solchen statischen Bedingungen vernünftig entfaltet?

Ein anderes beliebtes Open-Source-Projekt mit kommerziellen Wurzeln stützt unsere Thesen. Als Netscape die Communicator-Quellen im Jahr 1998 (Cubranic und Booth 1999) veröffentlichte, wurde das Mozilla-Projekt geboren, das für viele Jahre ein Beispiel für ein unwartbares und deshalb erfolgloses Open-Source-Projekt war. Heutzutage ist Mozilla grundlegend anders als der Communicator und das Projekt hat viele Veränderungen in der Architektur durchlebt, die auch zu erfolgreichen kommerziellen Anwendungen der neu entstandenen Mozilla-Komponenten geführt haben. Abbildung 3 zeigt, wie die Komplexität dieses einst einzigen, riesigen Produktes syste-

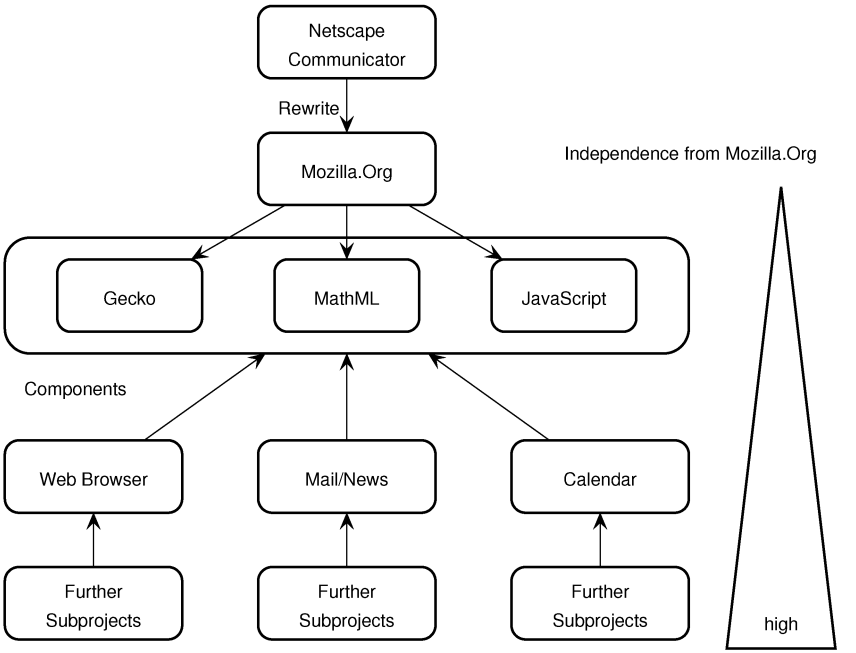


Abbildung 3: Interprojekt-Abhängigkeiten im Mozilla-Projekt

matisch in mehrere handhabbare Projekte aufgebrochen wurde, die mit dem weiteren Aufteilen des Projekts mehr und mehr an Flexibilität gewannen und unabhängiger von der Mozilla-Projektorganisation wurden.

Tatsächlich veröffentlichten im April 2003 die Mozilla-Projektleiter eine Erklärung, in der sie ihre bis dahin größte Aufteilung der Architektur ankündigten. Mozilla als solches wird nur als der Spitzname des Projekts weiterbestehen, während sich seine Kernkomponenten Mail, News und Web-Browser in separate Projekte verwandeln, wie sie die neue Roadmap widerspiegelt. Die Argumentation hinter diesen neuen Roadmap-Elementen lässt sich auf das Bevorzugen von Qualität statt Quantität zurückführen. Man muss sogar weniger tun, dafür aber besser und mit fehlerfreien Erweiterungsmechanismen. (*Mozilla Development Roadmap* 2003)

Momentan besteht Mozilla aus annähernd 50 „Kernprojekten" und über 2 Millionen Code-Zeilen, hat 13 Projektleiter und über 1 000 aktive Mitwirkende. Das Projekt hat sich letztendlich von seinen alten, nicht wartbaren Wurzeln befreit und floriert so sehr, dass Netscape in diesen Tagen seine Browser auf Mozilla basiert – anstatt umgekehrt.

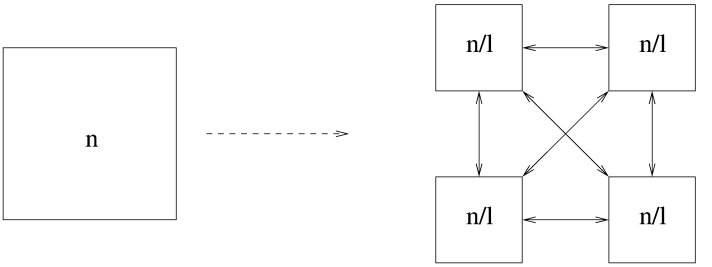


Abbildung 4: Code-Splitting

4.4. Komplexitätsreduktion durch Code-Aufteilung

Eine einfache, exemplarische Rechnung hilft, den Erfolg dieser Aufteilungsstrategie zu verstehen. Aus den Ergebnissen der Forschungsarbeiten zur Kostenschätzung, z. B. *function points* und COCOMO (Boehm 1981), ist die *Diseconomy of Scales* (Diversifikationsnachteil) wohl bekannt, das heißt, die Kosten eines Software-Projekts wachsen nicht linear mit dessen Größe, sondern wesentlich stärker.

Entsprechend dieser Erkenntnisse modelliert die COCOMO-Methode die Kosten eines Projekts mit Hilfe der Formel $A \cdot n^B$, wobei n ein Maß für die Größe des Systems ist (z. B. Quell-Code in KLOC – 1 000 Code-Zeilen) und $A, B > 1$ ist. Nun gehen wir von einem Stück Code aus, das aus n Modulen (ersetzbar durch KLOC) besteht. Weiter nehmen wir vereinfachend an, dass die Programmabhängigkeiten innerhalb dieses Programmfragments quadratisches Wachstum der Komplexität verursachen, d. h. $B = 2$. Dann kann die Gesamtkomplexität ausgedrückt werden durch folgende Formel:

$$C_{old} = O(n^2). \tag{1}$$

Wenn wir das Stück Code in l sorgfältig separierte Teile aufspalten, vermindert sich die Gesamtsystemkomplexität demnach auf

$$C_{new} = O\left(\frac{n^2}{l} + c \cdot l\right). \tag{2}$$

Mit anderen Worten: Die Komplexität C_{new} des gesamten Produkts wird signifikant reduziert. Die absolute Reduktion wächst sogar mit n und l. Die Konstante c ist ein linearer Faktor und repräsentiert die neuen Vermittlungskomponenten, die zwischen den nun getrennten l Teilen eingeführt werden müssen. Wegen der Entkopplung der l Teile können wir auch annehmen, dass die neuen Vermittler selbst keine quadratische Komplexität verursachen. Technisch wird dies durch den Gebrauch von schlanken Schnittstellen erreicht, wie sie von primitiven Operationen zur Interprozesskommunikation von Betriebssystemen oder gekapselten Datenstrukturen bereitgestellt werden.

Vom algorithmischen Standpunkt her bleibt die Komplexität – wenig überraschend – quadratisch in (1) und (2), aber in absoluten Zahlen wird C_{new} immer

kleiner sein als C_{old}, wenn n hinreichend groß gegenüber c ist. Die *Maintainer* von freien Softwareprojekten gewährleisten diese sinnvolle Teilung des Codes, sobald eine durchzuführende Änderung und die Verständlichkeit des Codes dies erfordern.

4.5. Inkrementelle Überarbeitungen

Erfolgreiche freie Softwareprojekte besitzen die Fähigkeit zur Durchführung inkrementeller Überarbeitungen von schwer zu wartenden Programmteilen. Mozilla[8] ist ein sehr gutes Beispiel für diese scheinbar gängige Praxis, denn Mozilla hat heute so gut wie keine Zeile Quell-Code mehr mit seinem Vorgänger Communicator gemeinsam, obwohl es niemals als Ganzes in einem einzigen Schritt neu aufgesetzt wurde.

Als kommerzielles Unternehmen machte die Firma Netscape allerdings einen schweren strategischen Fehler, als sie sich entschied, auf die vollständige Neuimplementierung der Code-Basis von Communicator zu warten, was letztendlich durch das Mozilla-Projekt geschah. Netscape war in dieser Zeit dazu gezwungen, die Version 5.0 des Communicators zu überspringen und hatte dadurch wertvolle Marktanteile hauptsächlich an den Internet Explorer von Microsoft verloren. Hätte Netscape die Prinzipien der erfolgreichen Evolution freier Software besser verstanden, so wäre dieser Schaden sicherlich abwendbar gewesen.

Ungeachtet der Verluste für Netscape hatte das Projekt Mozilla, das mit seiner freien und offenen Organisation keinem nennenswerten Marktdruck unterliegt, überwältigenden Erfolg. Im Rahmen von Mozilla sind eine ganze Reihe qualitativ hochwertiger Komponenten entstanden, die heute in vielen kommerziellen Anwendungen vorgefunden werden können (z. B. Borland Kylix API, AOL Web-Browser, Netscape Communicator, verschiedene eingebettete Web-Browser für Mobiltelefone und PDAs).

Dieser Fall zeigt die große Diskrepanz zwischen kommerzieller und freier Umgebung. In der freien Umgebung war das Vorgehen wegen der flexiblen, selbstgesteuerten Evolution sehr erfolgreich. In der kommerziellen Umgebung scheiterte das Vorhaben aufgrund fehlgeleiteter und unrealistischer Planung.

Wenn wir den Linux-Kernel als eine Neuimplementierung von AT&T UNIX, BSD oder MINIX betrachten, kommen wir zu ähnlichen Schlüssen: Noch vor einem Jahrzehnt wäre es für Firmen ein Desaster gewesen, das unreife Linux zu einem Baustein ihres Geschäftsmodells zu machen, aber für die freie Software-Gemeinde haben sich die letzten zehn Jahre intensiver Entwicklung als extrem erfolgreich herausgestellt. Und sogar im aktuellen Linux-Kern werden immer noch größere Teile schrittweise neu geschrieben. Ein aktuelles Beispiel dafür ist die Diskussion über die Restrukturierung des IDE-Layers.[9]

Die GCC-Suite, die sogar noch älter als Linux und Mozilla ist, unterliegt ebenso großen Veränderungen. Das Ziel des AST-Projekts[10] ist, die Handhabung der *abstract syntax trees* im Backend neu zu schreiben und dabei einen großen Teil der Optimierung von der Ebene der *register transfer language* auf die AST-Ebene anzuheben.

8 The Mozilla Project: http://www.mozilla.org/

9 siehe das Kernel Mailing List Archive http://www.uwsg.iu.edu/hypermail/linux/kernel/

10 Abstract Syntax Tree Optimizations http://www.gnu.org/software/gcc/projects/ast-optimizer.html

Hierzu gibt es Unterprojekte, wie den SSA-Zweig, der hauptsächlich von Red Hat gefördert wird, damit dieser verteilte und stufenweise Erneuerungsprozess zu einem Erfolg führt. Es ist offensichtlich, dass es keine triviale Aufgabe ist, ein 20 Jahre altes und gemächlich gewachsenes Backend eines Compilers zu restrukturieren. Frühere, erfolgreiche Überarbeitungen deuten jedoch auf die Tatsache hin, dass der Erfolg eines Open-Source-Projekts eng mit der Fähigkeit verbunden ist, sich selbst zu restrukturieren, wann und wo es notwendig wird. Wir gehen davon aus, dass dies ein weiteres wichtiges Prinzip der erfolgreichen Evolution langlebiger freier Softwareprojekte ist.

5. Fazit

Das erwiesenermaßen erfolgreiche Entwicklungsmodell freier Software ist eine ausgezeichnete Quelle für Prinzipien und Praktiken erfolgreicher Software-Evolution.

Im Gegensatz zu den meisten proprietären Softwareprojekten, ist die fortlaufende und uneingeschränkte Evolution ein inhärenter Bestandteil freier Software. Üblicherweise ist die einzige Konstante in einem freien Software-Projekt die ständige Veränderung. In Anbetracht Lehmans Gesetze der Software-Evolution (Lehman und Ramil 2001) ist es nicht überraschend, dass diese Strategie langlebige und qualitativ hochwertige Softwareprodukte hervorgebracht hat.

Der Prozess der Entwicklung freier Software ist alles andere als chaotisch sondern der Prozess und die Organisation skalieren mit der Größe des Projekts. Das heißt, Projekte starten fast ohne *overhead* und können rapide wachsen. Wenn jedoch eine bestimmte Größe überschritten wird, werden Regelungen, Lenkungsauschüsse und Werkzeuge je nach Bedarf hinzugefügt, d. h. der Prozess entwickelt sich. Die resultierende Organisation korreliert stark mit der technischen Struktur des Produkts und nicht mit der geographischen Verteilung der Teams oder dem Organigramm des Unternehmens.

Kommerzielle Software-Organisationen könnten aus der dynamischen Evolution von Prozessen, entsprechend dem Wachstum des technischen Produkts, ebenfalls großen Nutzen ziehen. Derzeit verwenden sie oft genau einen Prozess, der entweder starr in allen Projekten befolgt wird oder für die Projekte zuerst zugeschnitten werden muss. In der Regel gibt es keine Evolution des Prozesses wie bei freier Software, die die aktuellen Bedürfnisse des Projektes widerspiegelt. Obendrein schaffen es kommerzielle Softwareprojekte oft nicht, den richtigen Mitarbeitern die richtigen bzw. geeigneten Aufgaben zuzuweisen.

Natürlicher Wettbewerb und Auslese innerhalb von freien Software-Prozessen betonen eher fachliches Können als Autorität und Rang (in einem Unternehmen). Das erhöht die Qualität der Ergebnisse. Es ist sehr wahrscheinlich, dass eine Art von Wettbewerb kombiniert mit dynamischer Zuteilung von Rollen innerhalb von Unternehmen auch die Qualität von nicht-freien Softwareprodukten erhöhen würde. Natürlich würde das einen radikalen kulturellen Wechsel innerhalb der meisten Organisationen erfordern, aber dank agiler Methoden sind einige dieser Veränderungen bereits in kommerzielle Umgebungen diffundiert.

Neben der Evolution des Prozesses stehen einige der interessantesten Prinzipien freier Software in Zusammenhang mit der engen Verstrickung von einem sich ändernden Entwicklungsprozess und der Evolution der technischen Architektur, d. h. des Produkts selbst. Die Architektur des Produkts wird kaum vorausgeplant, sondern entwickelt sich frei mit den wechselnden Erfordernissen und mit der Größe des Produkts. Zu bestimmten Zeitpunkten werden die Architektur und Organisation in mehr oder weniger isolierte Teile aufgespalten, was zu unabhängig wartbaren Modulen oder sogar zu völlig neuen Produkten führt.

Die Entwicklung der Architektur wird von inkrementellen Überarbeitungen begleitet. Der Umfang der Überarbeitung wird allein durch die notwendige Änderung und die verfügbaren Ressourcen bestimmt. Im Gegensatz zu nicht-freien Umgebungen sind Überarbeitungen nicht durch nicht-technische Aspekte, wie mangelnde Rechte oder statische Verantwortlichkeiten, beschränkt. Das wiederum reduziert die Notwendigkeit für teure und ineffiziente *workarounds*, welche die Komplexität erhöhen und die Qualität und Wartbarkeit rapide sinken lassen. Freie Software entwickelt sich auf eine gesunde und natürliche Weise durch behutsames Erweitern und schonungslose inkrementelle Überarbeitung, bei der am Ende der am besten geeignete Code überlebt.

Die hier zusammengefassten Beobachtungen stützen unsere Anfangsthese, dass die freie Software-Bewegung Evolutionsstrategien hervorbringt, die weniger liberalen Umgebungen überlegen sind; ähnlich wie der Wirtschaftsliberalismus für die Ökonomie.

Bekanntmachung

Diese Arbeit wurde vom Deutschen Bundesministerium für Bildung und Forschung (BMBF) als Teil des Projekts ViSEK (Virtuelles Software- Engineering- Kompetenzzentrum) gefördert.

Literatur

Bauer, A. (2003), Compilation of Functional Programming Languages using GCC — Tail Calls, Master's thesis, Institut für Informatik, Technische Universität München. http://home.in.tum.de/~baueran/thesis/.

Beck, K. (1999), 'Embracing Change with Extreme Programming', *Computer* **32**, S. 70–77.

Boehm, B. (1981), *Software Engineering Economics*, Prentice-Hall.

Brooks Jr., F. P. (1995), *The Mythical Man-Month*, Addison Wesley.

Cederqvist, P. et al. (1993), *Version Management with CVS*, Signum Support AB.

Cubranic, D. und Booth, K. S. (1999), Coordinating open-source software development, *in* 'Eighth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises', IEEE Computer Society Press, Stanford, CA, USA, S. 61–65.

Darwin, C. (1859), *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, John Murray, London.

Godfrey, M. W. und Tu, Q. (2000), Evolution in Open Source Software: A Case Study, *in* 'Proceedings of the ICSM 2000', San Jose, CA, S. 131–142.

Harris, N. et al. (2002), *Linux Handbook / A guide to IBM Linux solutions and resources*, IBM International Technical Support Organization.

Intel Corporation (2001), *Intel Itanium Software Conventions and Runtime Architecture Guide*, Intel Corporation, Santa Clara, California. Intel document SC–2791, Rev. No. 2.4E.

Lehey, G. (2002), Evolution of a free software project, *in* 'Proceedings of the Australian Unix User's Group Annual Conference', Melbourne, Australia, S. 11–21.

Lehman, M. M. (1969), The Programming Process, techn. Bericht RC2722, IBM Research Centre, Yorktown Heights, NY.

Lehman, M. M. und Ramil, J. F. (2001), 'Rules and Tools for Software Evolution Planning and Management', *Annals of Software Engineering* .

Mozilla Development Roadmap (2003). http://www.mozilla.org/roadmap.html.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K. und Ye, Y. (2002), Evolution patterns of open-source software systems and communities, *in* 'Proceedings of the international workshop on Principles of software evolution', S. 76–85.

Opdyke, W. F. (1992), Refactoring Object-Oriented Frameworks, PhD thesis, University of Illinois at Urbana-Champaign.

Pizka, M. (1997), Design and Implementation of the GNU INSEL Compiler gic, Technical Report TUM I-9713, Technische Universität München.

Raymond, E. S. (1998), 'The Cathedral and the Bazaar'. Revision 1.40, http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Savoye, R. (2002), DejaGnu – The GNU Testing Framework, techn. Bericht, Free Software Foundation.

Smith, A. (1776), *The Wealth of Nations*, William Strahan, London.

Southern, J. und Murphy, C. (2002), 'Eine zielgerichtete Explosion', *Linux Magazin* .

Succi, G. und Eberlein, A. (2001), Preliminary Results from an Empirical Study on the Growth of Open Source and Commercial Software Products, *in* 'Proceedings of the Workshop on Economics-driven Software Engineering Research, EDSER 3', Toronto, Canada, S. 14–15.

The Standish Group International, I. (1995), 'CHAOS'.

The Standish Group International, I. (1999), 'CHAOS: A Recipe for Success'.

Wheeler, D. A. (2002), 'More Than a Gigabuck: Estimating GNU/Linux's Size', http://www.dwheeler.com/sloc/. Version 1.07.

van der Hoek, A. (2000), Configuration Management and Open Source Projects, *in* 'Proceedings of the 3rd International Workshop on Software Engineering over the Internet', Limerick, Ireland.

Snowbox

OLIVER FEILER

(GPL 2.0 siehe Seite 464)


```
#!/usr/bin/perl
# Snowbox - a simple POP3 server written in Perl
# Copyright 2004 Oliver Feiler <kiza@gmx.net>
# http://snowbox.kcore.de/
# 500 lines version

use strict;
use English;
use Digest::MD5 qw(md5_hex);
use Sys::Syslog qw(:DEFAULT setlogsock);
use Fcntl':flock';
use Fcntl;
use Socket;

my($snowbox_config) = "/etc/snowbox/config";  # change this if neccessary #
my($passwordfile) = "/etc/snowbox/user.auth";
my($maildrops) = "/var/mail/";
my($maildrop_gid) = "mail";
my($loglevel) = 1;
config();

my($logged_in) = 0;                    # Set to 1 after successful user login
my($input);                            # Network input
my($command);                          # POP3 command
my($argument);                         # POP3 command argument
my($argument2);                        # POP3 command argument
```

1 Begleitende Informationen befinden sich auf Seite 82, Abschnitt 4.

```
my($user);
my($pass);
my($num_messages);
my($maildrop_size);
my($myhostname) = `hostname`;
chomp($myhostname);
my($apop_stamp) = "<$PID.".time()."\@$myhostname>";
my(@maildrop);
my($remote) = "127.0.0.1";

if (getsockname(STDIN)) {
        $remote = inet_ntoa((unpack_sockaddr_in(getpeername(STDIN)))[1]);
}
$SIG{'HUP'} = 'signal_handler';       # Make sure to clean up if
$SIG{'INT'} = 'signal_handler';       # we get killed from a signal.
$SIG{'PIPE'} = 'signal_handler';
$| = 1;                               # make unbuffered
openlog ('snowbox', 'pid', 'user');   # open the syslog facility
setlogsock('unix');                   # specify unix socket for logging?
if ($loglevel >= 3) {
        syslog('debug', "connection from $remote.");
}

# Server starts
print "+OK Hi, nice to meet you. POP3 server ready $apop_stamp\r\n";
while (1) {
        $input = <STDIN>;
        $input =~ tr/\r\n//d;         # Remove trailing \r\n
        ($command,$argument,$argument2) = split(/ /,$input);
        if (!defined($command)) {
                pop_cmd_unknown();
                next;
        }
        if ((length($command) > 4) ||
                (length($argument) > 40) ||
                (length($argument2) > 40)) {
                pop_cmd_too_long();
                next;
        }
        $command =~ tr/a-z/A-Z/;        # Convert commands to uppercase
        $num_messages = 0;
        $maildrop_size = 0;
        foreach (@maildrop) {
                if (%$_->{"deleted"} == 0) {
                        $num_messages++;
                        $maildrop_size += %$_->{"len"};
                }
        }
        if ($command eq "USER") {
                if (!defined($argument)) {
                        pop_cmd_unknown();
                        next;
                }
                if ($logged_in == 1) {
                        print "-ERR Already logged in.\r\n";
                        next;
                }
                if (defined($user)) {
                        print "-ERR User name already given.\r\n";
                        next;
                }
                $user = $argument;
                $user =~ tr/a-zA-Z0-9\-\._//cd;      # Sanitize user name
                print "+OK May I have your password please?\r\n";
        }
```

```
elsif ($command eq "PASS") {
        if ($logged_in == 1) {
                print "-ERR Already logged in.\r\n";
                next;
        }
        $pass = $argument;
        auth_user("PLAIN", $user, $pass);
        $logged_in = 1;
        load_maildrop($user, "LOGIN");
}
elsif ($command eq "APOP") {
        if ($logged_in == 1) {
                print "-ERR Already logged in.\r\n";
                next;
        }
        $user = $argument;
        $pass = $argument2;
        $user =~ tr/a-zA-Z0-9\-\._//cd;      # Sanitize user name
        auth_user("APOP", $user, $pass);
        $logged_in = 1;
        load_maildrop($user, "LOGIN");
}
elsif ($command eq "AUTH") {
        if ($logged_in == 1) {
                print "-ERR Already logged in.\r\n";
                next;
        }
        pop_cmd_auth();
}
elsif ($command eq "STAT") {
        if (!$logged_in) {
                pop_cmd_not_logged_in();
                next;
        }
        pop_cmd_stat();
}
elsif ($command eq "LIST") {
        if (!$logged_in) {
                pop_cmd_not_logged_in();
                next;
        }
        pop_cmd_list($argument);
}
elsif ($command eq "RETR") {
        if (!$logged_in) {
                pop_cmd_not_logged_in();
                next;
        }
        pop_cmd_retr($argument);
}
elsif ($command eq "DELE") {
        if (!$logged_in) {
                pop_cmd_not_logged_in();
                next;
        }
        pop_cmd_dele($argument);
}
elsif ($command eq "NOOP") {
        pop_cmd_noop();
}
```

```
        elsif ($command eq "RSET") {
                if (!$logged_in) {
                        pop_cmd_not_logged_in();
                        next;
                }
                @maildrop = ();
                load_maildrop($user, "RSET");
        }
        elsif ($command eq "QUIT") {
                pop_cmd_quit($logged_in);
        }
        elsif ($command eq "UIDL") {
                if (!$logged_in) {
                        pop_cmd_not_logged_in();
                        next;
                }
                pop_cmd_uidl($argument);
        }
        else {
                pop_cmd_unknown();
        }
}


sub auth_user {
        my($sysuser);                           # Username in passwordfile
        my($syspass);                           # Password in passwordfile
        my($digest);
        my($access_granted) = 0;
        my($uid);
        my($gid);
        my(@pwdarray);
        my($auth) = shift;
        my($user) = shift;
        my($pass) = shift;
        if (!open (AUTHFILE, "$passwordfile")) {
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          could not open authorization file for reading.");
                }
                die "-ERR Internal error.
                  Could not read authorization file. \r\n";
        }
        while (<AUTHFILE>) {
                chomp;
                if (/^#/) {
                        next;
                }
                /^(.*?)\s+(.*)$/;
                $sysuser = $1;
                $syspass = $2;
                if ($user eq $sysuser) {
                        if ($auth eq "APOP") {
                              $digest = md5_hex($apop_stamp.$syspass);
                              if ($pass eq $digest) { $access_granted = 1; }
                        } elsif ($auth eq "PLAIN") {
                              if ($pass eq $syspass) { $access_granted = 1; }
                        }
                }
        }
        close (AUTHFILE);
```

```
        if (!$access_granted) {
                print "-ERR Login incorrect.\r\n";
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          failed login for \'$user\' using $auth.");
                }
                exit(1);
        }
        if ($loglevel >= 3) {
                syslog('debug', "connection from $remote:
                  \'$user\' logged in successfully using $auth.");
        }
        # Change to uid of logged user and gid of mail spool
        @pwdarray = getgrnam($maildrop_gid);
        $gid = $pwdarray[2];
        @pwdarray = getpwnam($user);
        $uid = $pwdarray[2];

        $GID = $gid;
        $EGID = "$gid $gid";
        if ($loglevel >= 3) {
                syslog('debug', "connection from $remote:
                  changed gid to \'$EGID\'.");
        }
        $UID = $uid;
        $EUID = $uid;
        if ($EUID != $uid) {
                syslog('warning', "connection from $remote: Internal error.
                  Could not change uid to \'$uid\'. Not good.");
                print "-ERR Internal error. setuid() failed. Not good.\r\n";
                exit(2);
        }
        if ($loglevel >= 3) {
                syslog('debug', "connection from $remote:
                  changed uid to \'$EUID\'.");
        }
}


sub load_maildrop {
        my($user) = shift;
        my($state) = shift;
        my($line) = "";
        my($blankline) = 1;     # First mesg is not preceded by a blank line
        my($count) = 1;
        $num_messages = 0;
        $maildrop_size = 0;
        if (!open (MAILDROP, "+<$maildrops/$user"))  {
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          mailbox for \'$user\' does not exist.");
                }
                die "-ERR Mailbox for this user does not exist.\r\n";
        }
        if ($state eq "LOGIN") { acquire_locks(); }
        while (<MAILDROP>) {
                if ((/^From /) && ($blankline == 1)) {
                        $blankline = 0;
                        $num_messages++;
                        if ($line ne "") {
                                my(%msghash);
                                $maildrop_size += length($line);
                                $msghash{"body"} = $line;
                                $msghash{"deleted"} = 0;
                                $msghash{"num"} = $count;
```

```
                                $msghash{"len"} = length($line);
                                push(@maildrop, \%msghash);
                                $count++;
                        }
                        $line = "";
                }
                $line = $line.$_;
                if ($_ eq "\n") {
                        $blankline = 1;
                } else {
                        $blankline = 0;
                }
        }
        if ($num_messages >= 1) {
                my(%msghash);
                $maildrop_size += length($line);
                $msghash{"body"} = $line;
                $msghash{"deleted"} = 0;
                $msghash{"num"} = $count;
                $msghash{"len"} = length($line);
                push(@maildrop, \%msghash);
        }
        print "+OK Welcome $user, you have $num_messages messages
          ($maildrop_size octets)\r\n";
}


# POP3 commands
sub pop_cmd_auth {
        print   "+OK I know the following authentication methods:\r\n".
                        "APOP\r\n".
                        ".\r\n";
}

sub pop_cmd_dele {
        my($message) = shift;
        $message =~ tr/0-9//cd;                 # Must only contain 0-9
        if (check_valid_msg_num($message) == -1) {
                return;
        }
        if ($message <= scalar(@maildrop)) {
                if ($maildrop[$message-1]->{"deleted"} == 0) {
                        $maildrop[$message-1]->{"deleted"} = 1;
                        print "+OK message $message deleted.\r\n";
                } else {
                        print "-ERR message $message already deleted.\r\n";
                }
        } else {
                print_err_no_such_msg();
        }
}
```

```
sub pop_cmd_list {
        my($message) = shift;
        if (!defined($message)) {
                print "+OK $num_messages messages
                  ($maildrop_size octets)\r\n";
                foreach (@maildrop) {
                        # Do not include deleted messages in listing
                        if (%$_->{"deleted"} == 0) {
                                print %$_->{"num"}." ".%$_->{"len"}."\r\n";
                        }
                }
                print ".\r\n";
        } else {
                $message =~ tr/0-9//cd;          # Must only contain 0-9
                if (check_valid_msg_num($message) == -1) {
                        return;
                }
                if (($message <= scalar(@maildrop)) &&
                    ($maildrop[$message-1]->{"deleted"} == 0)) {
                        print "+OK $message "
                          .$maildrop[$message-1]->{"len"}."\r\n";
                } else {
                        print_err_no_such_msg();
                }
        }
}

sub pop_cmd_noop {
        print "+OK *sighs idly*\r\n";
}

sub pop_cmd_quit {
        my($logged_in) = shift;
        if ($logged_in) {
                # Write maildrop
                seek (MAILDROP, 0, 0);
                truncate (MAILDROP, 0);
                foreach (@maildrop) {
                        if (%$_->{"deleted"} == 0) {
                                print MAILDROP %$_->{"body"};
                        }
                }
                nuke_locks();
        }
        print "+OK POP3 server signing off. Have a nice day.\r\n";
        closelog();                              # Close the syslog
        exit(0);
}
```

```
sub pop_cmd_retr {
        my($message) = shift;
        my($msg);
        if (!defined($message)) {
                print_err_no_such_msg();
                return;
        }
        $message =~ tr/0-9//cd;                 # Must only contain 0-9
        if (check_valid_msg_num($message) == -1) {
                return;
        }
        if ($message <= scalar(@maildrop)) {
                if ($maildrop[$message-1]->{"deleted"} == 1) {
                        print "-ERR message was deleted.\r\n";
                        return;
                }
        }
        if ($message <= scalar(@maildrop)) {
                print "+OK ".$maildrop[$message-1]->{"len"}." octets\r\n";
                $msg = $maildrop[$message-1]->{"body"};
                $msg =~ s/\n/\r\n/g;      # Replace all line endings with CRLF
                $msg =~ s/\r\n\.(?=\r\n)/\r\n\.\./g; # Prepad .CRLF -> ..CRLF
                print $msg;
                print ".\r\n";
        } else {
                print_err_no_such_msg();
        }
}

sub pop_cmd_stat {
        print "+OK $num_messages $maildrop_size\r\n";
}


sub pop_cmd_uidl {
        my($message) = shift;
        if (!defined($message)) {
                print "+OK Just a second, UIDL listing follows.\r\n";
                foreach (@maildrop) {
                        if (%$_->{"deleted"} == 0) {
                                print %$_->{"num"}." "
                                  .md5_hex(%$_->{"body"})."\r\n";
                        }
                }
                print ".\r\n";
        } else {
                $message =~ tr/0-9//cd;         # Must only contain 0-9
                if (check_valid_msg_num($message) == -1) {
                        return;
                }
                if (($message <= scalar(@maildrop)) &&
                    ($maildrop[$message-1]->{"deleted"} == 0)) {
                        print "+OK ".$maildrop[$message-1]->{"num"}." "
                          .md5_hex($maildrop[$message-1]->{"body"})."\r\n";
                } else {
                        print_err_no_such_msg();
                }
        }
}
```

```
sub pop_cmd_unknown {
        print "-ERR I have no idea what you want from me.\r\n";
        if ($loglevel >= 2) {
                syslog('info', "connection from $remote:
                  client sent unknown command.");
        }
}


sub check_valid_msg_num {
        my($message) = shift;
        if (($message eq "") || ($message < 1)) {
                print_err_no_such_msg();
                return -1;
        }
        return 0;
}


sub pop_cmd_not_logged_in {
        print "-ERR You are not logged in.\r\n";
        if ($loglevel >= 2) {
                syslog('info', "connection from $remote:
                  client sent transaction command while not logged in.");
        }
}

sub pop_cmd_too_long {
        print "-ERR Command too long.\r\n";
        if ($loglevel >= 1) {
                syslog('warning', "connection from $remote:
                  command sent was too long.");
        }
}


sub print_err_no_such_msg {
        print "-ERR No such message.\r\n";
}


sub acquire_locks {
        my($mtime);
        if (!flock (MAILDROP, LOCK_EX | LOCK_NB)) {
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          could not lock mailbox for \'$user\' (flock).");
                }
                die "-ERR Mailbox is locked.
                  Is another POP3 session active?\r\n";
        }
        # see if there is a dotlock. try to delete it if older than 15 mins
        if (-e"$maildrops/$user.lock") {
                $mtime = (stat("$maildrops/$user.lock"))[9];
                if (time()-$mtime > 900) {
                        if ($loglevel >= 1) {
                                syslog('warning', "connection from $remote:
                                  dotlock for \'$user\' is older than 15 min,
                                  ignoring.");
                        }
                        unlink ("$maildrops/$user.lock");
                }
        }
```

```
        # also set a dotlock here in case other programs ignore flock.
        if (!sysopen (DOTLOCK,
              "$maildrops/$user.lock", O_CREAT | O_RDWR | O_EXCL)) {
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          could not lock mailbox for \'$user\' (dotlock).");
                }
                die "-ERR Mailbox is locked.
                  Is another POP3 session active?\r\n";
        }
        print DOTLOCK $PID;
        close (DOTLOCK);
}


sub nuke_locks {
        flock (MAILDROP, LOCK_UN);
        unlink ("$maildrops/$user.lock");
        close (MAILDROP);
}

# Remove the locks if we get killed from inetd (SIGPIPE?)
sub signal_handler {
        if ($logged_in) {
                nuke_locks();
                if ($loglevel >= 2) {
                        syslog('info', "connection from $remote:
                          connection aborted, removing locks.");
                }
        } else {
                if ($loglevel >= 2) {
                        syslog('info', "connection from $remote:
                          connection aborted.");
                }
        }
        exit(2);
}


sub config {
        open (CONFIG, "$snowbox_config") || return; # and use default config
        while (<CONFIG>) {
                chomp;
                if (/^#/) { next; }
                /^(.*?)\s+(.*)$/;
                if ($1 eq "authfile") {
                        $passwordfile = $2;
                } elsif ($1 eq "maildir") {
                        $maildrops = $2;
                } elsif ($1 eq "maildir_gid") {
                        $maildrop_gid = $2;
                } elsif ($1 eq "loglevel") {
                        $loglevel = $2;
                }
        }
        close (CONFIG);
}
```

Die industrielle Nutzung und Entwicklung von Open-Source-Software: Embedded Linux

JOACHIM HENKEL UND MARK TINS

(CC-Lizenz siehe Seite 463)

Unternehmen tragen in stark steigendem Maße zur Entwicklung von Open–Source-Software (OSS) bei. Einen besonderen Fall von kommerzieller OSS–Entwicklung stellt „Embedded Linux" dar, also Varianten von Linux, die an den Gebrauch in „eingebetteten Systemen", wie z. B. Maschinensteuerungen und Handys, angepasst sind. Während Unternehmen wie IBM und Sun OSS aus strategischen Gründen und als Komplement zu ihren Produkten unterstützen, stellt *Embedded Linux* für Gerätehersteller einen festen Bestandteil ihres Produktes dar, für spezialisierte Softwareunternehmen sogar ihr Kerngeschäft. Daher wird die Entscheidung, eigene Entwicklungen zu *Embedded Linux* öffentlich zu machen, wesentlich vorsichtiger und selektiv getroffen. Es wird beschrieben, welche Schutzmöglichkeiten Unternehmen trotz der General Public License zur Verfügung stehen und in welchem Ausmaß Code trotz dieser Schutzmöglichkeiten freigegeben wird. Eine Analyse der Motive für eine Freigabe zeigt, dass externe Entwicklungsunterstützung tatsächlich als wichtiger Vorteil wahrgenommen wird. Für Softwareunternehmen, vor allem kleinere, spielt zudem der Marketingaspekt eine Rolle. Als begünstigend für OSS-Prozesse erweist sich die Tatsache, dass der Technologiebedarf im Bereich *Embedded Linux* extrem heterogen ist. Reines Trittbrettfahren ist daher kaum möglich; vielmehr müssen Unternehmen fast immer gewisse Anpassungen und Weiterentwicklungen bestehenden Codes vornehmen. Die rapide Verbreitung von *Embedded Linux* hat Konsequenzen für die gesamte Branchenstruktur der eingebetteten Betriebssysteme. Zum einen wird auch auf Anbieter proprietärer Betriebssysteme Druck ausgeübt, ihren Kunden den Quellcode verfügbar zu machen. Zum anderen reduziert *Embedded Linux* die Markteintrittsbarrieren, so dass eine zunehmende Fragmentierung des Marktes zu erwarten ist. Zusammenfassend lässt sich feststellen, dass trotz des weitgehend kommerziellen Umfeldes – Hobby-Entwickler spielen nur eine sehr untergeordnete Rolle – OSS-Entwicklungsprozesse im Bereich *Embedded Linux* sehr gut funktionieren.

1. Einführung

Open-Source-Software (OSS) hat sich von einem Nischenphänomen für Hobby-Entwickler zum Mainstream entwickelt. Neben Hobby-Programmierern sind große wie kleine Softwareunternehmen an der Entwicklung von OSS beteiligt und zunehmend auch Unternehmen außerhalb der Softwarebranche. Um einen solchen Fall von OSS geht es in diesem Beitrag: um „Embedded Linux". Dieser Begriff bezeichnet Varianten des OSS-Betriebssystems, die an den Gebrauch in „eingebetteten Systemen" wie z. B. Maschinensteuerungen oder Mobiltelefonen angepasst sind.

Die Situation bei *Embedded Linux* unterscheidet sich deutlich von der des Standard-Linux, dem beispielsweise IBM Unterstützung zuteil werden lässt: Während es für IBM vor allem um das strategische Ziel geht, Linux möglichst weit zu etablieren, stellt *Embedded Linux* für Gerätehersteller einen festen Bestandteil ihrer Produkte dar, für darauf spezialisierte Softwarehersteller sogar ihr Kerngeschäft. Die Entscheidung, Code öffentlich zu machen und sich aktiv an OSS-Projekten zu beteiligen, erscheint daher in ganz anderem Licht. Dazu kommt, dass Hardwareunternehmen typischerweise noch weniger Erfahrung mit der OSS-Kultur und OSS-Prozessen haben als Softwareunternehmen. Für Gerätehersteller bedeutet aktive Teilnahme an OSS-Projekten quasi eine Offenlegung von Teilen ihres Innovationsprozesses – ein für Manager äußerst ungewöhnlicher, sogar absurder Gedanke.

Inwiefern OSS-Prozesse in diesem Umfeld dennoch funktionieren, wurde mit verschiedenen Forschungsansätzen untersucht. Trotz der Bestimmungen der General Public License (GPL), unter der Linux lizenziert ist, bestehen durchaus verschiedene Schutzmöglichkeiten. Es zeigt sich, dass Unternehmen diese Möglichkeiten nutzen und etwa die Hälfte ihrer Entwicklungen zu *Embedded Linux* schützen während sie die andere Hälfte freigeben. Als wichtigste Motive für eine Freigabe werden externe Entwicklungsunterstützung sowie, von Softwareunternehmen, Marketing angegeben. Dabei erweist sich ein Charakteristikum von *Embedded Linux* – die hohe Heterogenität im Technologiebedarf und damit auch in Softwareentwicklungen – als sehr förderlich für den OSS-Entwicklungsprozess.

Der Artikel ist wie folgt aufgebaut: In Abschnitt 2. werden die Forschungsprojekte vorgestellt – Interviews und eine großzahlige Befragung – aus denen die hier dargestellten Resultate stammen. Abschnitt 3. vermittelt technische Informationen zu *Embedded Software* allgemein und *Embedded Linux*. In Abschnitt 4. wird die Entwicklung von *Embedded Linux* beschrieben: wer dazu beiträgt, aus welchen Gründen, was begünstigende Faktoren sind und welche Schutzmöglichkeiten trotz der GPL bestehen. Abschnitt 5. schließt den Beitrag mit einem Ausblick ab.

2. Forschungsmethodik

2.1. Interviews

Im Zeitraum von Mai 2002 bis Juni 2003 wurden von einem der Autoren insgesamt 30 Interviews geführt. Unter den Gesprächspartnern waren Vertreter von 13 Soft-

wareunternehmen, die auf die Entwicklung von *Embedded Linux* spezialisiert sind, Vertreter von sechs Geräteherstellern, die Linux in ihren Geräten verwenden und, zumindest zum Teil, auch selbst entwickeln sowie sieben Branchenexperten, die entweder über *Embedded Linux* schreiben oder in anderer Weise einen relevanten Bezug dazu haben. Um auch eine externe Perspektive zu berücksichtigen, wurden zudem vier Gespräche geführt mit Vertretern von Softwareunternehmen, die nur indirekt mit *Embedded Linux* zu tun haben, in den meisten Fällen als Anbieter von Konkurrenzprodukten. Geographisch befinden sich die Befragten zu etwa gleichen Teilen in den USA (17 Interviews) und Europa (13 Interviews).

Unter den Gesprächspartnern fanden sich die meisten der als wichtig angesehenen in auf *Embedded Linux* spezialisierten Softwareunternehmen. Dazu zählen in den USA FSMLabs, Lineo (inzwischen Teil von MetroWerks), LynuxWorks, MontaVista und TimeSys, in Deutschland Denx Software und Sysgo. Außerdem wurden kleinere Unternehmen bzw. Einzelentwickler befragt (zwei in den USA, vier in Europa).

Gerätehersteller spielen bei *Embedded Linux* einerseits als Nachfrager von Entwicklungen bei Softwareunternehmen eine Rolle (Auftragsentwicklung), andererseits auch selbst als Entwickler. Auch hier finden sich Unternehmen verschiedener Größe unter den Interviewpartnern – Siemens ebenso wie kleinere Unternehmen, z. B. Innominate und SSV Embedded Systems.

Unter den befragten Branchenexperten finden sich drei Autoren, drei Unternehmen bzw. Einzelpersonen, die mit Standard-Linux arbeiten sowie ein Vertreter der Free Software Foundation. Da deren Gründer Richard Stallman die GPL entworfen hat, unter der auch *Embedded Linux* steht, war vor allem die Einschätzung der Geschäftspraktiken der Embedded-Linux-Unternehmen durch die Free Software Foundation interessant.

Zur Abrundung des Bildes und zum besseren Verständnis der technischen und marktlichen Grundlagen wurden schließlich auch drei Vertreter von Unternehmen befragt, die proprietäre eingebettete Betriebssysteme entwickeln. Mit Wind River, Microsoft und QNX wurden dabei die wichtigsten Unternehmen auf diesem Markt ausgewählt.

Die Interviews wurden mit Personen geführt, die mit der zugrunde liegenden Technik und vor allem dem Entwicklungsprozess von *Embedded Linux* vertraut sind. In 28 Fällen handelte es sich um mündliche Interviews von im Mittel 53 Minuten Dauer, in zwei Fällen um eine Befragung per E-Mail. Weitere Details zum Vorgehen finden sich bei Henkel (2003).

2.2. Großzahlige Befragung

Basierend auf den Erkenntnissen der interviewbasierten Untersuchung wurde ein Fragebogen konzipiert, um ein quantitatives Bild der Entwicklung von *Embedded Linux* zu erhalten. Zwischen November 2003 und März 2004 war der Fragebogen online, der auf geeigneten Web-Portalen und Mailinglisten angekündigt wurde. Die Zahl der gültigen Rückläufe betrug 268. Vollständige deskriptive Statistiken der Daten finden sich bei Henkel und Tins (2004).

Die Teilnehmer der Befragung erwiesen sich als sehr erfahrene Programmierer, mit im Mittel 14,2 Jahren Erfahrung in der Softwareentwicklung. Die durchschnittliche Erfahrung mit der Entwicklung von OSS und *Embedded Software* betrug 4,9 bzw. 7,1 Jahre.

Von den 268 Teilnehmern arbeiten 197 für kommerzielle Unternehmen, 30 für Universitäten und 41 als Hobby-Entwickler. Unter den Unternehmen finden sich Software-, Geräte- und Komponentenhersteller. Darunter sind Softwareunternehmen im Mittel am jüngsten und kleinsten: Der Median des Gründungsjahres beträgt 1997, 64 % der Befragten arbeiten für „kleine" Unternehmen mit maximal 50 Mitarbeitern. Gerätehersteller nehmen eine mittlere Position ein (Median 1988: 54 % kleine Unternehmen), Komponentenhersteller sind im Mittel am ältesten und größten (Median 1986: 26 % kleine Unternehmen).

3. Technische Aspekte

3.1. Embedded Software

Ein PC oder ein Großcomputer sind dazu vorgesehen, dass auf ihnen eine Vielzahl verschiedener Softwareprogramme laufen. Im Gegensatz dazu wird „embedded software" oder „eingebettete Software" in einem Gerät verwendet, das einem bestimmten Zweck gewidmet ist. Neben der Zweckgebundenheit der Hardware-Software-Kombination ist ein weiteres typisches Merkmal solcher „eingebetteter Systeme" eine eingeschränkte Benutzerschnittstelle. Wenn überhaupt eine Schnittstelle existiert, kann diese beispielsweise nur aus einigen Schaltern, einem kleinen Bildschirm oder einem Touchscreen bestehen (Lombardo 2001, S. xv). Eine scharfe Abgrenzung zwischen eingebetteter und anderer Software ist allerdings schwierig (Hollabaugh 2002, S. 8).

Die Verbreitung eingebetteter Systeme nimmt rapide zu. Schon 2001 wurde der Anteil der Mikroprozessoren, die in eingebetteten Systemen anstatt in „normalen" Computern verbaut werden, auf über 80 Prozent geschätzt (Lehrbaum 2001). Beispiele für eingebettete Systeme sind PDAs, Telefone, Videospielkonsolen, Küchengeräte, Geräte der Mess- und Regeltechnik, Netzwerk-Router, Maschinensteuerungen, Kraftfahrzeuge, Flugzeuge und Kraftwerke.

Diese Beispiele illustrieren die große Vielfalt eingebetteter Systeme, die sich in einer hohen Heterogenität der verwendeten Prozessoren und Komponenten widerspiegelt. Diese Heterogenität spielt für die Entwicklung von *Embedded Linux* eine wichtige Rolle (siehe Abschnitt 4.3.). Während für PCs Intels X86-Prozessorarchitektur dominiert, werden in eingebetteten Systemen verschiedenste Prozessoren eingesetzt, darunter ARM, ColdFire, MIPS, PowerPC, Sparc, Super H, X86, XScale sowie Microcontroller (Dankwardt 2003). Entsprechend fragmentiert ist die Software-Branche im Bereich eingebetteter Systeme. So hat z. B. Microsoft, anders als bei PC-Betriebssystemen, bei eingebetteten Betriebssystemen zwar eine starke, aber durchaus keine dominante Stellung. Als Marktführer wird gemeinhin Wind River genannt, aber auch Unternehmen wie z. B. QNX spielen eine wichtige Rolle. Zudem ist Eigenentwicklung

durch den Gerätehersteller eine wichtige Quelle eingebetteter Betriebssysteme, wenn auch ihre Bedeutung abnimmt.

Die verschiedenen Verwendungen eingebetteter Systeme implizieren stark heterogene Anforderungen an die eingebettete Software. Als typische Anforderungen können Stabilität, geringer Speicherplatzbedarf und Echtzeit-Fähigkeit genannt werden. Echtzeit-Fähigkeit beispielsweise bedeutet, dass das System auf Signale mit einer deterministischen Antwortzeit reagiert (Dankwardt 2002*a*), die zudem oft kurz sein muss. Diese Eigenschaft ist bei PC-Betriebssystemen nicht gegeben und z. B. bei Maschinensteuerungen und Flugzeugen von essentieller Bedeutung.

3.2. Embedded Linux

Es existiert keine „offizielle Version" von *Embedded Linux*. Yaghmour (2003) polemisiert sogar:

> „Let's put it bluntly: Embedded Linux doesn't exist. Embedded Linux is the stuff of glitzy announcements, hype, and other marketing mumbo jumbo."

Sein Kritikpunkt ist, dass es kein spezielles „Embedded Linux" gibt – vielmehr wird auch in eingebetteten Systemen im Wesentlichen Standard-Linux verwendet, den jeweiligen Anforderungen entsprechend angepasst und erweitert. Dennoch wird in diesem Beitrag, wie allgemein üblich, „Embedded Linux" als Bezeichnung für Konfigurationen bzw. Varianten von Linux verwendet, die in gewisser Weise spezifisch für den Einsatz in eingebetteten Systemen sind. In den meisten Fällen liegt diese Spezifität darin begründet, dass zusätzliche Module integriert werden, die die besonderen Anforderungen eingebetteter Systeme erfüllen, und dass nicht benötigter Code entfernt wird.

Wie erläutert, sind eingebettete Systeme ausgezeichnet durch eine sehr spezifische Aufgabe, große Heterogenität und typischerweise die Anforderung nach hoher Stabilität, Echtzeit-Fähigkeit und geringem Speicherplatzbedarf. Entsprechende Anpassungen werden bei Standard-Linux vorgenommen, um es für den Einsatz in eingebetteten Systemen verwendbar zu machen.

Die Spezifität der Aufgabe bringt es mit sich, dass ein Großteil des Codes von Standard-Linux nicht benötigt wird und deshalb entfernt werden kann. Heterogenität der Hardware macht es notwendig, Linux auf verschiedene Prozessor-Architekturen und verschiedene Boards zu portieren. Dabei zählen die Portierungen auf die PowerPC- und die ARM-Prozessoren zu den wichtigsten Projekten im Bereich *Embedded Linux*.

Zur Erhöhung der Stabilität gibt es keine speziellen Module, wohl aber ein Vorgehen, das im Server-Bereich sehr ungewöhnlich wäre: Es wird teilweise auf alte Kernel-Versionen zurückgegriffen, die zwar nicht alle Funktionalitäten der aktuellen Version haben, sich jedoch durch sehr hohe und erprobte Stabilität auszeichnen. So wurde in einem Interview, im Herbst 2002, von der Verwendung der Kernel-Version 2.4.4 vom Mai 2000 berichtet.

Zur Verbesserung der Echtzeit-Fähigkeit existieren mehrere Ansätze. In RTLinux läuft Linux als Aufgabe niedrigster Priorität eines darunter liegenden Echtzeit-Kernels (Dankwardt 2002*b*). Ein alternativer Ansatz wird vom RTAI-Projekt (Realtime Application Interface) verfolgt, und MontaVista gab im Oktober 2004 den Start eines Open-Source-Projektes bekannt, das „harte" Echtzeit für Linux ermöglichen soll (LinuxDevices.com 2004).

Ein geringer Speicherplatzbedarf wird zum einen dadurch erreicht, dass nicht benötigter Code entfernt wird. Zum anderen existieren eine Reihe von Tools, die speziell diesem Bedürfnis Rechnung tragen, so z. B. die C-Bibliothek uClibc und die Tool–Sammlung BusyBox.

Ein wichtiger technischer Aspekt ist, dass neue Versionen von *Embedded Linux* nicht durch Fortentwicklung alter Versionen entstehen, sondern neu aus dem jeweils aktuellen Standard-Kernel abgeleitet werden. Damit bleibt Kompatibilität weitgehend gewahrt und, noch wichtiger, profitiert *Embedded Linux* von der technischen Weiterentwicklung der Standardversion.

Die Verwendung von Linux in eingebetteten Systemen ist noch relativ jung. In der Befragung ergab sich eine stark linksschiefe Verteilung der Erfahrung, die die Unternehmen mit *Embedded Linux* haben: Nur sehr wenige (12,5 %) starteten mit der Arbeit an *Embedded Linux* in den sechs Jahren zwischen 1994 (dem frühesten Datum) und 1999, das Gros dagegen in den vier Jahren zwischen 2000 und 2003. Für Softwareunternehmen in der Stichprobe liegt der mittlere Start-Zeitpunkt signifikant früher (um 9 Monate) als für Hardwareunternehmen.

Heute ist *Embedded Linux* zu einer wichtigen Technologie im Markt für eingebettete Betriebssysteme geworden (Webb 2002). Einer Untersuchung des Marktforschungsunternehmens VDC zufolge ist es sogar das derzeit am häufigsten gewählte Betriebssystem für eingebettete Systeme, vor Windows und VxWorks von Wind River (VDC 2004). Aktivitäten wie die Bildung des „CELF" (Consumer Electronics Linux Forum) durch Matsushita, Sony, Hitachi, IBM, NEC, Philips, Samsung, Sharp und Toshiba im Juni 2003 unterstreichen die inzwischen sehr breite Akzeptanz des Open-Source-Betriebssystems in der Industrie.

Diese Attraktivität hat verschiedene Gründe: Aufgrund der wachsenden technischen Anforderungen an eingebettete Betriebssysteme (graphische Benutzeroberflächen, Netzwerkfähigkeit) sind vollständig selbstentwickelte Systeme der Gerätehersteller oft keine adäquate Lösung mehr. *Embedded Linux* erlaubt, auf Basis eines etablierten und stabilen OSS-Betriebssystems nur diejenigen Anpassungen als Eigenentwicklungen vorzunehmen oder von Auftragsentwicklern vornehmen zu lassen, die für die jeweilige Anwendung erforderlich sind. Diese Anpassungen werden erleichtert dadurch, dass Linux in hohem Maße modular aufgebaut ist (Baldwin und Clark 2003). Neben diesen technischen Aspekten ist natürlich vorteilhaft, dass *Embedded Linux* als OSS keine Stücklizenzgebühren erfordert. Zudem vermeidet die Lizenzierung als OSS die Transaktionskosten, die bei der Lizenzierung proprietärer Betriebssysteme typischerweise entstehen und die Entwicklung erheblich bremsen können. Ein Interviewpartner berichtete von einem Fall, in dem ein Unternehmen zum Betrieb eines Internet-Radiosenders eine bestimmte proprietäre Software lizenzieren wollte:

> „That was the slowest part of their process! They had to go out and negotiate a whole separate license. Whether or not you choose free software for a project depends largely on how much transaction cost you are willing to put up with. The more free software, the faster the development process."

4. Entwicklung von Embedded Linux

4.1. Beiträge zur Entwicklung

In den Medien wird OSS oft als Software dargestellt, die von antikommerziell eingestellten Hobby-Programmierern entwickelt wird. Dieses Bild trifft für OSS heutzutage schon im Allgemeinen kaum zu; für *Embedded Linux* ist es sogar völlig falsch. In diesem Bereich stammt der Großteil der Entwicklungsbeiträge von Unternehmen.

In der Befragung wurden die Teilnehmer gebeten, anzugeben, für welche Art von Organisation sie arbeiten. Dabei waren fünf Alternativen vorgegeben. Die meisten klassifizierten ihr Unternehmen als Hardware-Hersteller (51,1 %), darunter 42,5 % Gerätehersteller und 8,6 % Komponentenhersteller (z. B. Chips, Boards). Etwa ein Viertel der Befragten arbeitet für Softwareunternehmen (22,4 %), ein weiteres Viertel im nicht-kommerziellen Bereich (26,5 %). Davon wiederum entfallen 15,3 % auf Hobby-Entwickler, 11,2 % auf Universitäten.

Diese Zahlen geben nur ein eingeschränktes Bild davon, woher die Entwicklungsbeiträge zu *Embedded Linux* anteilig stammen: Aufgrund der Selbstselektion der Befragten liegt keine repräsentative Stichprobe vor. Man kann jedoch vermuten, dass Teilnehmer aus dem nicht-kommerziellen Sektor überrepräsentiert sind, da sie eine höhere emotionale Anteilnahme und geringeren Zeitdruck haben dürften. Demnach ist der reale Anteil an Hobby-Entwicklern wohl noch deutlich geringer als 15 %. Dazu kommt, dass diese wesentlich weniger Stunden pro Woche an *Embedded Linux* arbeiten als kommerzielle Entwickler. Diese Resultate bestätigen, dass Hobby-Entwickler für *Embedded Linux* nur eine untergeordnete Rolle spielen. Allerdings ist erwähnenswert (und Gegenstand laufender Forschungsarbeiten), dass 60 % der befragten kommerziellen Entwickler angaben, auch in ihrer Freizeit an *Embedded Linux* zu arbeiten, wenn auch zumeist nur wenige Stunden pro Woche. Motivationen ähnlich denen der „typischen" OSS-Hobby-Entwickler scheinen also auch in diesem Bereich eine Rolle zu spielen.

Um die Herkunft des öffentlich verfügbaren Codes für *Embedded Linux* noch näher zu ergründen, wurden die Teilnehmer gefragt, wie viel Code ihre Unternehmen öffentlich machen (siehe zu folgenden Angaben Henkel 2004*b*). Genauer: Welcher Anteil des Codes für *Embedded Linux*, den das Unternehmen entwickelt und der nicht zu spezifisch ist (also potentiell nützlich ist für andere), wird auf Mailinglisten oder zum Download auf Websites zugänglich gemacht? Die durchschnittlichen Prozentangaben unterscheiden sich deutlich für kommerzielle (49 %) und nicht-kommerzielle (92 %) Entwickler bzw. Organisationen. Unter den kommerziellen geben Software-Unternehmen und Gerätehersteller mit im Mittel 58 % bzw. 59 % relativ

viel frei, während Gerätehersteller nur 42 % freigeben. Aber auch innerhalb dieser Gruppen variiert das Freigabeverhalten sehr stark: In jeder der drei kommerziellen Kategorien finden sich Unternehmen, die nur 1 % freigeben, aber auch solche, die 100 % öffentlich machen. Eine multivariate Analyse des Anteils freigegebenen Codes erlaubt es, verschiedene Einflussfaktoren zu identifizieren. Darunter finden sich die Erfahrung des Unternehmens mit *Embedded Linux* (positiver Einfluss) sowie die Größe des Unternehmens (negativer Einfluss).

4.2. Motive für Unternehmen, Code freizugeben

Es ist inzwischen alltäglich, dass Unternehmen zur Entwicklung von OSS beitragen. Wichtige Beispiele dafür sind IBM, Sun, Red Hat und SuSE. Die Motive kommerzieller Unternehmen, ihre Softwareentwicklungen als OSS frei zugänglich zu machen, sind verschiedentlich untersucht und beschrieben worden, insbesondere unter dem Stichwort „OSS Business Models" (Behlendorf 1999, Hecker 1999, Raymond 1999, Feller und Fitzgerald 2002, Lerner und Tirole 2002, Wichmann 2002, Bonaccorsi und Rossi 2004, Dahlander 2004).

Als wichtige Motive werden von diesen Autoren die folgenden genannt: Etablierung eines Standards und Herstellen von Kompatibilität, Erhöhung der Nachfrage nach komplementären Gütern und Dienstleistungen, Generierung externer Entwicklungsunterstützung, Anpassung bestehender OSS an die spezifischen Bedürfnisse des Unternehmens sowie Signalisieren von technischer Leistungsfähigkeit und/oder Beachtung der OSS-Normen (Osterloh et al. 2001, Franck und Jungwirth 2003).

Für die Entwicklung von *Embedded Linux* sind einige dieser Motive nur von geringem Belang. Diese Software ist ein fester Teil der Produkte, der eingebetteten Systeme, die die Gerätehersteller anbieten. Kompatibilität auf der Ebene des Betriebssystems ist daher, beispielsweise bei Maschinensteuerungen, oft nur von geringer Bedeutung. Auch die Erhöhung der Nachfrage nach komplementären Produkten spielt hier keine Rolle, und externe Entwicklungsunterstützung zumindest von *Hobby*-Entwicklern ist, wie oben dargestellt, kaum zu erwarten. Umgekehrt erschwert die GPL, unter der auch *Embedded Linux* steht, den Schutz des „geistigen Eigentums", das im Code enthalten ist (siehe dazu 4.4.). Was also sind *hier* die Motive von Unternehmen, ihre Entwicklungen öffentlich verfügbar zu machen?

Basierend auf den Interviews wurden den Teilnehmern der Befragung zwölf mögliche Gründe vorgegeben, warum ihr Unternehmen Code zu *Embedded Linux* freigibt. Sie wurden gebeten, ihre Zustimmung auf einer 5-Punkte-Skala von „stimme völlig zu" bis „stimme gar nicht zu" anzugeben. Das Ausmaß von Zustimmung und Ablehnung ist in Abbildung 1 wiedergegeben. Die Abbildung zeigt nur Antworten von Befragten, die bei Hardwareunternehmen arbeiten. Eine Differenzierung zwischen Hardware- und Softwareunternehmen ist notwendig, da sich die relative Bedeutung der Gründe zwischen den beiden Gruppen deutlich unterscheidet. Auf Softwareunternehmen wird unten eingegangen.

Die höchste Zustimmung erhielt die Aussage „My company reveals code because the GPL requires it." Dies war zu erwarten, da die GPL einen offensichtlich starken

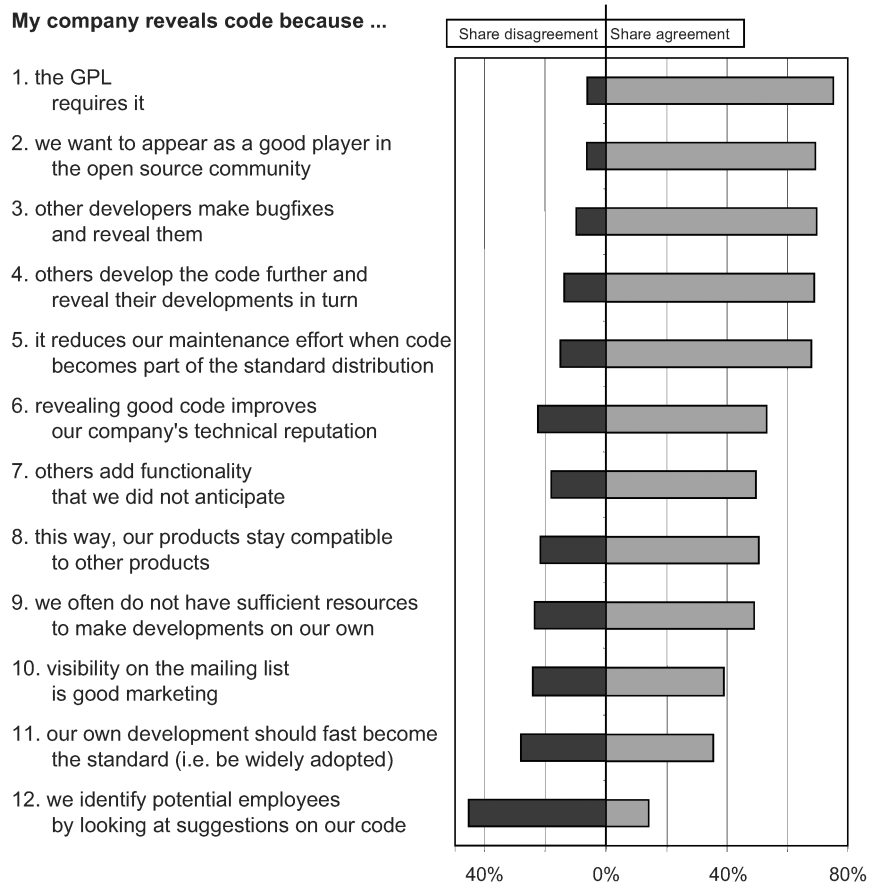


*Abbildung 1: Anteil von Zustimmung und Ablehnung in der Befragung zu verschiedenen Gründen, Code für Embedded Linux zu veröffentlichen (nur Entwickler bei Hardwareunternehmen, N=137, Quelle: Henkel 2004*b*)*

Einfluss auf das Freigabeverhalten hat. Die auf den Rängen 2 bis 5 folgenden Gründe erhielten ebenfalls sehr hohe Zustimmung. Drei davon – „bugfixes by others", „further development by others" sowie „reduced maintenance effort" – beziehen sich direkt auf informelle externe Entwicklungsunterstützung, also auf die *technischen Vorteile* des Open-Source-Prozesses. Der zweitplazierte Grund („to appear as a good OSS player") hat ebenfalls (indirekt) damit zu tun, da Entwicklungsunterstützung aus einer OSS-Community – bestehe sie aus privaten oder kommerziellen Entwicklern – nur erwarten kann, wer sich als „guter OSS-Player" erwiesen hat. Es zeigt sich also, dass auch in diesem weitgehend kommerziellen Bereich der OSS-Entwicklungspro-

zess funktioniert und technische Vorteile mit sich bringt, und dass Unternehmen aus diesem Grund bereit sind, Code öffentlich zu machen. Auf Rang 6, mit signifikant geringerer Zustimmung als Rang 5, folgt ein Grund, der mit Marketing zu tun hat: „revealing good code improves our company's technical reputation." Weitere Details können der Abbildung entnommen werden.

Marketing als Grund für die Freigabe von Code ist aus nahe liegenden Gründen für Softwareunternehmen wesentlich wichtiger, da diese zumeist als Zulieferer für Hardwareunternehmen arbeiten, d. h. als Auftragsentwickler oder Anbieter von Tools oder spezifischen Distributionen. Entsprechend finden sich Marketing-bezogene Gründe bei Softwareunternehmen auf höheren Rangplätzen: "revealing good code improves our company's technical reputation" auf Rang 3, "visibility on the mailing list is good marketing" auf Rang 6.

Die Bedeutung, als guter OSS-Player wahrgenommen zu werden, wird auch durch folgendes Fallbeispiel belegt. Die Software RTLinux, die Echtzeitfähigkeit von Linux ermöglicht, wird vom Unternehmen FSMLabs gepflegt, dem auf die Technologie ein Patent erteilt wurde (US-Patent Nr. 5.995.745). Das Patent ist aus zwei Gründen umstritten. Zum einen besteht ein relativ breiter Konsens, dass das Patent nicht hätte erteilt werden dürfen, da der Ansatz, ein umfangreiches Betriebssystem auf einem kleinen Echtzeit-Kernel laufen zu lassen, nicht neu ist. Zweitens wird FSMLabs vorgeworfen, sich durch die Patentierung Entwicklungen der „RTLinux-Community" widerrechtlich angeeignet zu haben. Als Reaktion auf die Erteilung dieses Patents (und auf die unklare Haltung von FSMLabs zu Lizenzbedingungen) wanderten sehr viele Teilnehmer der RTLinux-Community zum Konkurrenzprojekt RTAI ab, das ausschließlich auf OSS aufbaut und keine patentgeschützten Elemente enthält.

4.3. Heterogenität im Bedarf

Wie in Abschnitt 3.2. beschrieben, steht der Begriff „Embedded Linux" nicht für ein genau definiertes Softwarepaket, sondern für eine Vielfalt von Anpassungen und Erweiterungen, die Linux für den Einsatz in eingebetteten Systemen geeignet machen. Diese Vielfalt rührt her aus der starken Heterogenität des Technologiebedarfs: Je nach Einsatzfeld stehen z. B. Echtzeitfähigkeit, Stabilität, geringer Speicherplatzbedarf, ungewöhnliche Peripherie oder die Anpassung an besondere Prozessorarchitekturen im Vordergrund.

Ein deutscher Gerätehersteller bemerkte dazu im Frühjahr 2003:

> „So wie ich das einschätze, gibt es halt keinen Mainstream-Weg [. . .] Es gibt in diesem Bereich eigentlich nicht mehr so den großen *Kernel-Maintainer* für die eingebetteten Systeme, und da finden Sie einfach wirklich den totalen Wildwuchs und die totale Vielfalt in Abhängigkeit von den Prozessorplattformen."

Das heißt, es gibt eine große Zahl von Projekten, die nicht Teil des Standard-Kernels sind, sondern als eigenständige Projekte gepflegt werden. Diese Vielfalt kann für den

Anwender die Qual der Wahl bedeuten, wird jedoch als sehr attraktiv angesehen. Der gleiche Interviewpartner stellt fest:

> „Was sich bei uns dramatisch geändert hat, ist, dass es seit 2000 viele Lösungen speziell für den Embedded-Markt gibt, Module. [. . .] Das war für uns der entscheidende Punkt, dann da drauf zu switchen."

Heterogenität hat aber nicht nur Konsequenzen für Anwender, die bestehenden OSS-Code nutzen, sondern auch und vor allem für die Weiterentwicklung von *Embedded Linux*. (Die folgende Diskussion gibt verkürzt einige der Ergebnisse von Henkel (2004*a*) wieder.) Trotz der Vielfalt ist oft nicht *genau* die Lösung verfügbar, die ein Entwickler benötigt, so dass eigene Programmierarbeit notwendig wird:

> „The embedded market is so extremely fragmented that no solution fits all needs. That is, the demand for specific adaptations is enormous" (Europäischer Gerätehersteller).

Damit wird ein Problem behoben, das in der Spieltheorie als „waiting game" bezeichnet wird: Verschiedene Akteure benötigen ein bestimmtes Modul, aber keiner will die notwendige Entwicklungsarbeit auf sich nehmen – jeder wartet also darauf, dass ein anderer Akteur die Entwicklung vornimmt. Eine solche Situation tritt nicht auf, wenn der Technologiebedarf stark heterogen ist, da es unwahrscheinlich ist, dass ein anderer Akteur *genau* denselben Bedarf hat. Ein Interviewpartner aus einem US-Softwareunternehmen, spezialisiert auf *Embedded Linux*, stellt dazu fest:

> „We're very much customer-driven. If the customer needs something and it's not available in the open source, we'll just do it. And we're not going to wait for somebody else to do it. I think everybody else sees that about the same way."

Noch ein weiteres Problem wird durch die Heterogenität im Technologiebedarf behoben, zumindest aber reduziert. Die Freigabe von Code stellt in dem Maße einen Verlust an Wettbewerbsvorteilen dar, in dem enge Wettbewerber diesen Code unverändert nutzen können. Heterogenität im Bedarf bedeutet jedoch tendenziell, dass der Wettbewerb weniger hart ist (da die Angebote differenziert sind); zudem können Wettbewerber den Code zumeist eben nicht unverändert nutzen, wie oben dargelegt.

Natürlich besteht ein Trade-off zwischen Schutz durch Spezifität und Interesse der Community an breiter verwendbarem Code. Eine *extrem* spezifische Entwicklung, die nur in einem ganz bestimmten Gerät sinnvoll eingesetzt werden kann, kann weitgehend ohne Verlust von Wettbewerbsvorteilen öffentlich gemacht werden; allerdings wird sie für andere Entwickler kaum interessant sein, so dass die potentiellen Vorteile des OSS-Entwicklungsprozesses nicht realisiert werden. Eine Freigabe ist daher sinnvoll für Entwicklungen mittlerer Spezifität – diese können für andere von Nutzen sein, jedoch aufgrund anderer technischer Schwerpunkte von geringerem Nutzen als für den Entwickler selbst. Eine Unterscheidung ist schließlich auch sinnvoll zwischen *differenzierendem* spezifischen Code, der wettbewerbsrelevante Merkmale verkörpert, und

solchem, der allein aus notwendiger Anpassung an bestimmte Bauelemente spezifisch ist. Eine ausführlichere Diskussion findet sich bei Henkel (2003, 2004*a*).

Heterogenität im Technologiebedarf wurde auch bei der Webserver-Software Apache gefunden und hatte auch dort positive Auswirkungen auf OSS-Entwicklungen (Franke und von Hippel 2003). Aufgrund der viel stärkeren Heterogenität der Hardware im Bereich *Embedded Linux* liegt es nahe, dass der positive Effekt auf den OSS-Prozess hier noch ausgeprägter ist. Zusammenfassend bleibt also festzuhalten, dass hohe Heterogenität im Technologiebedarf ein zentrales Merkmal ist, das *Embedded Linux* von Standard-Linux unterscheidet und das wichtige – und positive – Implikationen für den OSS-Entwicklungsprozess von *Embedded Linux* hat.

4.4. Schutzmöglichkeiten

Im obigen Text wurde mehrfach von den Vor- und Nachteilen gesprochen, die eine Freigabe von Code zu *Embedded Linux* für das entwickelnde Unternehmen hat. Diese Diskussion setzt voraus, dass es eine gewisse *Entscheidungsfreiheit* hinsichtlich Freigabe oder Schutz gibt. Glaubt man jedoch den (vereinfachenden) Darstellungen der GPL, die gelegentlich in der Presse zu finden sind, scheint diese Entscheidungsfreiheit nicht zu bestehen: Demnach *muss* Code, der auf bestehender OSS unter der GPL basiert (also *derived work* im Sinne der GPL darstellt), öffentlich gemacht werden. Diese Interpretation der GPL ist jedoch falsch. Die Lizenz besagt in diesem Zusammenhang lediglich, dass jeder, der im rechtmäßigen Besitz des kompilierten Codes eines unter der GPL lizenzierten Programms ist, auch Anspruch auf den Quellcode hat sowie das Recht, diesen zu verändern und weiter zu verbreiten.

Für die Entwicklung von *Embedded Linux* haben die Bestimmungen der GPL folgende Konsequenzen: Weiterentwicklungen und Anpassungen des Betriebssystems sind *derived work*, müssen also wiederum unter der GPL lizenziert werden. Ein Softwareunternehmen, das Auftragsentwicklungen für einen Gerätehersteller vornimmt, ist demnach verpflichtet, diesem den Quellcode zugänglich zu machen – durchaus keine ungewöhnliche Forderung, eher der Normalfall. Auch Anbieter proprietärer eingebetteter Betriebssysteme stellen ihren Kunden den Quellcode zur Verfügung, wenn auch typischerweise nur gegen Gebühr und unter Einschränkungen (siehe z. B. Windowsfordevices.com 2004).

Stärkere Implikationen als für ein Softwareunternehmen hat die GPL für einen Gerätehersteller, der verpflichtet ist, jedem Käufer des Gerätes den Quellcode der darin enthaltenen Version von *Embedded Linux* zugänglich zu machen. Soweit es sich um ein preisgünstiges Gerät für den Massenmarkt handelt, ist der Code damit quasi öffentlich verfügbar, insbesondere für Wettbewerber.

Aber auch in diesem Fall existieren Schutzmöglichkeiten. Von der Entwicklung des Codes bis zur Markteinführung des Gerätes vergehen im Mittel etwa 18 Monate – ein beträchtlicher Vorsprung für das entwickelnde Unternehmen. Zudem muss der Code, wie schon erwähnt, nur den Käufern des Gerätes zugänglich gemacht werden. Wenn diese relativ wenige sind und sie außerdem kein Interesse am Quellcode haben (man denke z. B. an Bauunternehmen, die Kräne der Firma Liebherr samt dem darin

enthaltenen Code von *Embedded Linux* kaufen), kann der Code erfolgreich geheim gehalten werden. Schließlich ist es im Bereich *Embedded Linux* gängige Praxis, wenn auch umstritten, Treiber nur als *binary loadable modules* freizugeben, also ohne ihren Quellcode (Marti 2002).

Trotz der Auflagen der GPL bestehen also durchaus einige Schutz- und Geheimhaltungsmöglichkeiten für Weiterentwicklungen von *Embedded Linux*. Unternehmen haben demnach in der Tat Entscheidungsfreiheiten: Code kann direkt nach seiner Entstehung einem geeigneten öffentlichen OSS-Projekt über dessen Mailingliste zugänglich gemacht werden oder auf der Website des Unternehmens zum Download angeboten werden. Alternativ kann das Unternehmen die Schutzmöglichkeiten maximal ausnutzen und den Code erst so spät wie möglich und an so wenige Empfänger wie möglich herausgeben. Eine empirische Untersuchung (Henkel 2004*b*) zeigt, dass Unternehmen die jeweiligen Vor- und Nachteile von Geheimhaltung und Offenheit bewusst abwägen. Diese Möglichkeit ist zweifellos von hoher Bedeutung für die weite und schnell wachsende Verbreitung von *Embedded Linux*.

5. Ausblick

Embedded Linux ist dabei, die Branche der eingebetteten Betriebssysteme erheblich zu verändern. Für Gerätehersteller, die bisher selbstentwickelte Systeme verwendet haben, stellt es einen attraktiven Mittelweg zwischen vollständiger Eigenentwicklung und komplett externer Beschaffung dar. Einige weitere Implikationen sollen im Folgenden angesprochen werden.

Eine wichtige Auswirkung betrifft den Grad der Offenheit. Wettbewerb zwischen Betriebssystemen findet in verschiedenen Dimensionen statt, darunter Preis, Leistungsfähigkeit und eben auch Offenheit. Offenheit, also die Verfügbarkeit des Quellcodes, ist für Gerätehersteller und Anwendungsentwickler äußerst nützlich, da dadurch die Fehlersuche stark vereinfacht wird. Das Beispiel von *Embedded Linux* hat den Anwendern gezeigt, dass Offenheit durchaus möglich ist – und als Konsequenz daraus ist eine stärkere Nachfrage nach Offenheit entstanden. Sehr deutlich zeigt sich dies in der empirischen Untersuchung bei Komponentenherstellern: 92 % der Befragten antworteten, dass ihr Unternehmen zum Zeitpunkt der Befragung (Ende 2003 bis Anfang 2004) mehr Code freigibt als im Jahr 2000; 8 % stellten keine Änderung fest. Dabei handelt es sich typischerweise um Treiber für die Komponenten, die das Unternehmen anbietet. Diese Steigerung deckt sich mit Aussagen eines Geräteherstellers, für den die Verfügbarkeit des Treiber-Quellcodes ein wichtiges Kriterium für die Auswahl der Komponente darstellt. Der Markt legt zunehmend Wert auf offenen Quellcode und macht Kaufentscheidungen davon abhängig. Einen weiteren Beleg für den von OSS und *Embedded Linux* ausgelösten „Trend zu Offenheit" stellt die weitergehende Öffnung des Quellcodes eingebetteter Versionen von Windows dar (Windowsfordevices.com 2004).

Eine zweite Auswirkung dürfte sein, dass die Fragmentierung der Branche zunimmt. Zwar war der Markt für eingebettete Betriebssysteme aufgrund der Heterogenität im Technologiebedarf auch bisher schon wesentlich fragmentierter als der für z. B. Server-

oder Desktopbetriebssysteme. Wegen der stark steigenden Anforderungen (beispielsweise im Hinblick auf Netzwerkfähigkeit und graphische Oberflächen) ist jedoch eine Konsolidierung abzusehen: Die Kosten, die die laufende Weiterentwicklung eines proprietären Betriebssystems verursacht, steigen stark an. Diese Kosten entstehen bei *Embedded Linux* nur in geringem Maße, da ein stabiles, leistungsfähiges und ständig weiterentwickeltes Betriebssystem frei verfügbar ist. Ein Unternehmen kann sich daher darauf beschränken, Anpassungen an spezifische Kundenwünsche anzubieten. Markteintrittsbarrieren werden somit deutlich reduziert, was tendenziell eine stärkere Fragmentierung der Branche nach sich zieht. Und nicht nur die Verfügbarkeit von OSS-Code, auch der OSS-Entwicklungsprozess begünstigen kleine Unternehmen und damit eine Fragmentierung (Gruber und Henkel 2004): Zum einen vereinfacht der OSS-Entwicklungsprozess die Tätigkeit ressourcenbeschränkter kleiner Unternehmen, da in gewissem Maße externe Entwicklungsunterstützung verfügbar ist. Zum anderen bieten Mailinglisten eine kostengünstige und aussagekräftige Marketingplattform gerade für kleine Softwareunternehmen.

Das Beispiel von *Embedded Linux* zeigt, dass Open Source und Kommerzialisierung keinen Widerspruch darstellen, auch nicht im industriellen Kontext. Unternehmen müssen sich in gewissem Maße umstellen – die freiwillige Offenlegung eigener Entwicklungen ist äußerst unüblich und widerspricht lang geübter Praxis. Sie kann jedoch, wie die Untersuchungen gezeigt haben, für Unternehmen sehr vorteilhaft sein.

Literaturverzeichnis

Baldwin, C. Y. und Clark, K. B. (2003), 'The Architecture of Cooperation: How Code Architecture Mitigates Free Riding in the Open Source Development Model', Working Paper, Harvard Business School.

Behlendorf, B. (1999), Open Source as a Business Strategy, *in* C. Dibona, S. Ockman und M. Stone (Hrsg.), 'Opensources: Voices from the Open Source Revolution', O'Reilly, Sebastopol, CA, S. 149–170.

Bonaccorsi, A. und Rossi, C. (2004), 'Comparing motivations of individual programmers and firms to take part in the open source movement. From community to business', Working paper, Sant'Anna School of Advanced Studies Pisa.

Dahlander, L. (2004), 'Appropriating returns from open innovation processes: A multiple case study of small firms in open source software', Working Paper, Chalmers University of Technology, Gothenburg. http://opensource.mit.edu/papers/dahlander.pdf.

Dankwardt, K. (2002*a*), 'ELJonline: Real Time and Linux, Part 1', LinuxDevices.com, http://www.linuxdevices.com/articles/AT5997007602.html [28. Aug 2003].

Dankwardt, K. (2002*b*), 'ELJonline: Real Time and Linux, Part 3', LinuxDevices.com, http://www.linuxdevices.com/articles/AT6320079446.html [28. Aug 2003].

Dankwardt, K. (2003), 'Update on "Where's the free beer?"', LinuxDevices.com, http://www.linuxdevices.com/articles/AT8733630838.html [19. Mai 2003].

Feller, J. und Fitzgerald, B. (2002), *Understanding Open Source software development*, Addison Wesley, Boston, MA.

Franck, E. und Jungwirth, C. (2003), 'Reconciling investors and donators – The governance structure of open source', *Journal of Management and Governance* **7**, S. 401–421.

Franke, N. und von Hippel, E. (2003), 'Satisfying Heterogeneous User Needs via Innovation Toolkits: The Case of Apache Security Software', *Research Policy* **32**(7), S. 1199–1215.

Gruber, M. und Henkel, J. (2004), 'New ventures based on open innovation – an empirical analysis of start-up firms in embedded Linux', Working Paper, University of Munich. http://opensource.mit.edu/papers/gruberhenkel.pdf.

Hecker, F. (1999), 'Setting Up Shop: The Business of Open-Source Software', *IEEE Software* **16**(1), S. 45–51.

Henkel, J. (2003), Open-Source-Aktivitäten von Unternehmen – Innovation in kollektiven Prozessen, Habilitationsschrift, Ludwig-Maximilians-Universität München.

Henkel, J. (2004*a*), 'The Jukebox Mode of Innovation – a Model of Commercial Open Source Development', CEPR Discussion Paper 4507, http://opensource.mit.edu/papers/henkel.pdf.

Henkel, J. (2004*b*), 'Patterns of free revealing – Balancing code sharing and protection in commercial open source development', Working Paper, University of Munich http://www.inno-tec.bwl.uni-muenchen.de/forschung/henkel/Henkel_PatternsOfFreeRevealing_2004-08.pdf.

Henkel, J. und Tins, M. (2004), 'Munich/MIT Survey: Development of embedded Linux', Working paper, University of Munich. http://opensource.mit.edu/papers/henkeltins.pdf.

Hollabaugh, C. (2002), *Embedded Linux: Hardware, Software, and Interfacing*, Addison-Wesley, Boston.

Lehrbaum, R. (2001), 'Developer Interest in Embedded Linux Skyrockets', LinuxDevices.com, http://www.linuxdevices.com/articles/AT2492406168.html [28. Aug 2003].

Lerner, J. und Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), S. 197–234.

LinuxDevices.com (2004), 'MontaVista unveils "open" hard-real-time Linux project', http://www.linuxdevices.com/news/NS3557180455.html.

Lombardo, J. (2001), *Embedded Linux*, New Riders, Boston, MA.

Marti, D. (2002), 'Use Binary-Only Kernel Modules, Hate Life', LinuxJournal.com vom 19. Jun 2002, http://www.linuxjournal.com/article.php?sid=6152 [01. Sep 2003].

Osterloh, M., Rota, S. und von Wartburg, M. (2001), 'Open Source – New Rules in Software Development', Working Paper, Institut für betriebswirtschaftliche Forschung, Universität Zürich, http://www.iou.unizh.ch/orga/downloads/OpenSourceAoM.pdf.

Raymond, E. S. (1999), *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, 1. Aufl., O'Reilly, Sebastopol, CA.

VDC (2004), 'White paper, Venture Development Corporation', Referenced in: "Linux now top choice of embedded developers", LinuxDevices.com, http://www.linuxdevices.com/news/NS2744182736.html.

Webb, W. (2002), 'Pick and Place: Linux Grabs the Embedded Market', *EDN.COM* **10/31/2002**. http://www.edn.com/contents/images/253780.pdf.

Wichmann, T. (2002), 'FLOSS Final Report – Part 2: Free/Libre Open Source Software: Survey and Study – Firms' Open Source Activities: Motivations and Policy Implications', Berlecon Research, http://www.berlecon.de/studien/downloads/200207FLOSS_Activities.pdf.

Windowsfordevices.com (2004), 'Microsoft eases 'shared source' restrictions', http://www.windowsfordevices.com/news/NS8570590617.html.

Yaghmour, K. (2003), 'Embedded Linux: Semantics and Reality', O'Reilly Network, http://linux.oreillynet.com/pub/a/linux/2003/05/12/embedlinux.html [19. Mai 2003].

Kapitel 3

Ökonomie

„Leicht kann der Hirt eine ganze Herde Schafe vor sich hintreiben, der Stier zieht seinen Pflug ohne Widerstand; aber dem edeln Pferde, das du reiten willst, musst du seine Gedanken ablernen, du musst nichts Unkluges, nichts unklug von ihm verlangen."

– J. W. Goethe, Egmont (1789)

Die ökonomischen Dimensionen von Open Source

MATTHIAS BÄRWOLFF

(CC-Lizenz siehe Seite 463)

Die wirtschaftliche Bedeutung von Open Source ist heute weitestgehend unzweifelhaft. Ängste und Vorurteile weichen stetig nüchternen Wirtschaftlichkeitsbetrachtungen. So ist immer mehr Unternehmensberatungen und Marktforschungsinstituten bewusst, dass wir einen strukturellen Wandel in der Softwareindustrie erleben, der die Strategiehorizonte praktisch aller Firmen, nicht nur die der Softwareindustrie, berührt.

Dabei geht es nicht nur um Software, sondern nicht zuletzt auch um die Lehren aus der Open-Source-Bewegung. Immer mehr Firmen wird klar, dass Kunden mehr als nur passive Konsumenten sind, dass sie nämlich eine aktive Rolle bei der Schöpfung von Innovationen spielen können. Immer mehr Firmen wird auch klar, dass mit starren Return-on-Investment-Formeln aus dem Industriezeitalter heute keine nachhaltige Differenzierung im Wettbewerb erreicht werden kann und dass Transparenz, Beteiligung und die gezielte Aufgabe von exklusiver Kontrolle über Eigentum sich positiv auf langfristige Kundenbeziehungen auswirken können. Open Source zeigt, wo die wirtschaftlichen Potenziale in Bereichen liegen, die durch Industriestandards geprägt sind und in denen eine Differenzierung über Produktattribute praktisch unmöglich ist. Immer mehr Firmen verabschieden sich von einem streng proprietären Geschäftsmodell und spezialisieren sich zunehmend auf Beratung und Software-Anpassung im Bereich Open Source, selbst wenn sie die maßgebliche Kraft hinter einer Software sind, wie etwa MySQL AB mit der MySQL-Datenbank oder die Zope Corporation mit dem *Content Management Framework* (CMF) Zope.

Gerade auch die jüngere Vergangenheit zeigt, dass Open Source „erwachsen" geworden ist. Der Einfluss kommerzieller Interessen hat substantiell an Bedeutung gewonnen und damit auch die Relevanz deren ökonomischer Analyse. Während Ökonomen sich bislang vorwiegend mit der Motivation von Open-Source-Softwareentwicklern befassten, steht zunehmend die volkswirtschaftliche und institutionelle Bedeutung von Open Source als Wertschöpfungs- und Innovationsparadigma im Vordergrund ihrer Betrachtungen. War man noch bis vor kurzem der Meinung, dass kommerzielle Interessen die Motivation hinter Open Source gefährden könnten, scheint es immer mehr, dass kommerzielle Interessen durchaus positive Einflüsse auf Open–Source-Softwareprojekte haben können. Mithin sollte es darum gehen, die Vorteile aus „beiden Welten" zu vereinen, auch wenn die Dichotomie Open Source – Kommerzielle Software zunehmend verschwimmt und letztlich eigentlich irrelevant ist.

Linus Torvalds hat einmal treffend bemerkt, dass Open-Source-Software nicht die Lösung für globale Probleme wie Unterernährung sein kann und stellt damit nicht zuletzt Utopien von einer besseren Welt, basierend auf „Open-Source-Gesellschaften", vehement in Frage.

Und doch stellt sich die Frage nach der Reichweite von Open Source. Ist das Innovationsparadigma, wie manche Beobachter glauben, auf andere Industrien übertragbar? Zeigt es neue und nachhaltige Wege des Wirtschaftens auf? Oder verändert es die klassischen Wirtschaftsprozesse nur marginal? Diese und andere Fragen sind noch immer Gegenstand teilweise heftiger Debatten. Eines ist in jedem Fall klar: Open Source ist kein Phänomen mehr, sondern praktisch allgegenwärtig – In den Produktionszyklen der Softwareindustrie, in den Backoffices der Wirtschaft und zunehmend auf den Desktops der Anwender. Die Revolution wird zusehends zu einer Evolution, bei der nicht nur die Summe der individuellen Wohlfahrten steigt, sondern gewohnte Prozesse und Strukturen aufgebrochen werden, womit nicht zuletzt auch ein kultureller Wandel einhergeht – zweifellos zum Besseren, nicht zum Schlechteren.

Die Autoren dieses Kapitels versuchen, die angerissenen Fragestellungen von verschiedenen vorwiegend mikroökonomisch geprägten Perspektiven zu beleuchten und für den Leser verständlich aufzubereiten. Der Beitrag von Jens Mundhenke dient dabei als Einführung in die klassische ökonomische Betrachtung von Open Source als „Handelsgut" in Softwaremärkten. Dabei werden im Hinblick auf Open-Source–Software primär die Themen Monopolisierung im Bereich Wissensgüter sowie die Bestreitbarkeit von Monopolmärkten behandelt.

Der zweite Beitrag von Christian Maaß und Ewald Scherm widmet sich der aktuell äußerst lebhaft diskutierten Frage von Standards als Element der Wettbewerbsregulierung, sowohl auf Seiten der Produzenten und Konsumenten als auch auf Seiten des Gesetzgebers. Insbesondere wird eine kritische Unterscheidung zwischen Open–Source-Software und offenen Standards getroffen, die in vielen Debatten nur sehr unscharf erfolgt.

Im dritten Artikel dieses Kapitels greift Maik Hetmank die in der Literatur relativ rege bearbeitete Frage nach der Motivation einzelner Entwickler in Open-Source-Projekten auf. Dabei macht er durch die Analyse einer Vielzahl von empirischen Befunden deutlich, dass der „extrinsische" Motivationsfaktor eine größere Rolle spielt als in der populären Literatur gemeinhin angenommen.

Walter Seemayer und Jason Matusow von der Firma Microsoft widmen sich im vierten Beitrag der Shared-Source-Initiative ihres Unternehmens. Sie erörtern unter anderem die strategischen Potenziale von Open Source als Instrument für eine traditionell proprietär ausgerichtete Softwarefirma. Weiterhin stellen sie den Begriff des *Software-Ökosystems* zur Diskussion, mit dem sie die wechselseitigen Abhängigkeiten zwischen Privatwirtschaft, öffentlichem Sektor und nichtkommerziellen Privatinitiativen beleuchten und die Rolle von Microsoft in diesem System deutlich machen wollen.

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

JENS MUNDHENKE

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

1. Einleitung

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

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

2. Software als digitales Gut

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

2.1. Angebotsseitige Eigenschaften

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

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

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

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

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

2.2. Nachfrageseitige Eigenschaften

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

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

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

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

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

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

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

2.3. Implikationen für den Wettbewerb auf Softwaremärkten

Ausschließbarkeit als Voraussetzung für das Entstehen von Softwaremärkten

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

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

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

Tendenz zur Marktkonzentration und zur Bevorzugung etablierter Anbieter

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

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

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

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

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

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

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

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

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

Spektrum strategischer Instrumente verfügen.

Hohe Marktdynamik durch den Zwang zu Innovation

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

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

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

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

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

3. Wettbewerbs- und Innovationspotenziale von Open Source

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

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

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

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

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

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

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

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

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

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

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

3.2. Wettbewerbswirkungen

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

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

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

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

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

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

3.3. Innovationspotenziale von Open-Source-Software

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

Niedrige Markteintrittsbarrieren

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

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

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

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

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

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

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

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

Aufbau von Humankapital

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

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

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

Offene Schnittstellen

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

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

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

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

3.4. Falsche Innovationsanreize im Open-Source-Modell?

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

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

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

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

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

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

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

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

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

4. Zusammenfassung

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

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

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

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

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

Literaturverzeichnis

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

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

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

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

Ghosh, R. A., Glott, R., Krieger, B. und Robles, G. (2002), FLOSS Final Report - Part 4: Survey of Developers, *in* 'Free/Libre and Open Source Software: Survey and Study', International Institute of Infonomics, University of Maastricht and Berlecon Research GmbH. http://www.infonomics.nl/FLOSS/report/FLOSS_Final4.pdf.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Open-Source-Software und Standardisierung

Christian Maass und Ewald Scherm

Open-Source-Software steht seit geraumer Zeit im öffentlichen Interesse. Die Auseinandersetzung mit Fragen der Standardisierung erfolgt allerdings nur am Rande. Vielmehr werden damit zusammenhängende Probleme nur oberflächlich dargestellt, oder es kommt zu Fehleinschätzungen. So erachtet man OSS und offene Standards mitunter als Synonyme, obwohl es sich dabei um zwei grundlegend verschiedene Sachverhalte handelt. Vor diesem Hintergrund erfolgt in diesem Beitrag eine Auseinandersetzung mit drei Aspekten, die in Zusammenhang mit Fragen der Standardisierung zwar immer wieder angesprochen, aber nur oberflächlich dargestellt werden: (1) Kompatibilitätsprobleme in Folge des *Forking*, (2) Hersteller(un)abhängigkeit und (3) der Einfluss von Softwarepatenten. Die Auseinandersetzung mit diesen Aspekten soll dazu beitragen, die Diskussion um Open-Source-Software und damit einhergehende Fragen der Standardisierung, differenzierter führen zu können.

1. Ausgangssituation

In den vergangenen Jahren ist Open-Source-Software (OSS) zunehmend in das Interesse der Öffentlichkeit gerückt. Das liegt unter anderem daran, dass dem Anwender von OSS auf Grundlage einer Open-Source-Lizenz sehr weitgehende Nutzungsrechte eingeräumt werden, die im Gegensatz zum „traditionellen" Softwaregeschäft stehen. So kann OSS faktisch kostenlos aus dem Internet bezogen werden, und es ist für den Anwender möglich, den Quellcode – der Aufbau und Arbeitsweise eines Programms offen legt – einzusehen sowie Änderungen daran vorzunehmen. Außerdem ist es gestattet, den Quellcode weiterzugeben, wobei keine Lizenzgebühren erhoben werden dürfen.[1]

Vor diesem Hintergrund betrachtet man OSS als ein öffentliches Gut[2] (Kollock 1999, Bessen 2002, von Krogh 2003), das einen scheinbaren Widerspruch zu dem

1 Vgl. dazu Punkt 1, Free Distribution, der Open Source Definition (Version 1.9), online: http://www.opensource.org/docs/definition.php [25.11.2004].

2 Ein öffentliches Gut ist dadurch gekennzeichnet, dass Personen nicht vom Konsum ausgeschlossen werden können. Gleichzeitig schränkt der Güterkonsum eines Individuums nicht die Konsummöglichkeit anderer Marktteilnehmer ein. Aufgrund dieser Eigenschaften ist es für Unternehmen – die eine Maximierung ihrer Gewinne anstreben – nicht attraktiv, dieses Gut zu produzieren.

in der Ökonomie angenommenen Nutzenmaximierungskalkül darstellt (Franck und Jungwirth 2001). Beiträge, in denen das Phänomen OSS näher betrachtet wird, untersuchen vor allem die Kalküle derer, die OSS entwickeln. Dabei geht es um die Frage, warum Softwareentwickler unentgeltlich ihre Entwicklungsleistung in ein Open–Source-Projekt einbringen (Lerner und Tirole 2000, Hars und Ou 2001, Lakhani und Wolf 2003). Weiterhin betrachtet man den Ablauf der Entwicklung quelloffener Software und damit zusammenhängende organisatorische (Garzarelli 2002, Morner 2002, Demil und Lecocq 2003) und technische Fragen (Koch und Schneider 2002, Scacchi 2002, Shaikh und Cornfold 2003). In diesem Zusammenhang wird in dem Prinzip der quelloffenen Produktentwicklung auch ein neuer Ansatz zur Produktinnovation gesehen (Lerner und Tirole 2001, Chesbrough 2003, von Krogh 2003, Gfaller 2004). Nicht zuletzt ist die Rolle des Staates Objekt einer kritischen Analyse, da sich eine starke Präferenz der öffentlichen Hand für OSS feststellen lässt (Schmidt und Schnitzer 2002, Comino und Manenti 2003).

Die Auseinandersetzung mit Fragen der Standardisierung – worunter man eine Vereinheitlichung nach bestimmten Regeln oder Mustern versteht (Knorr 1993, S. 23) – erfolgt in der wissenschaftlichen Diskussion allerdings nur am Rande. Vielmehr werden damit zusammenhängende Probleme nur oberflächlich dargestellt oder es kommt zu Fehleinschätzungen. So werden OSS und offene Standards im Rahmen einer Informationsbroschüre des Wirtschaftsministeriums des Landes Baden-Württemberg weitestgehend als Synonyme betrachtet (Wirtschaftsministerium 2004, S. 7), obwohl es sich dabei um zwei grundlegend verschiedene Sachverhalte handelt: Während offene Standards darauf abzielen – unabhängig von einem Hersteller – Kompatibilität zwischen den verschiedenen Komponenten eines EDV-Systems herzustellen, geht es bei quelloffener Software um eine spezielle Form der Softwareentwicklung. Open–Source-Lizenzen wie die *GNU General Public Licence* (GPL) sehen dabei vor, dass der Weiterverbreitung des Quellcodes keine Restriktionen in Form von Exklusivitätsklauseln in Lizenzen oder Ansprüchen aus Patenten entgegenstehen, weshalb die Softwareentwicklung konsequenterweise auf offenen Standards beruht. Weiterhin findet man häufig die folgende Auffassung:

> „OSS can help a business or public institution avoid getting locked into a vicious circle of hardware and software upgrades and changes in data formats that require investing in new license fees and significant retraining and can provoke major down time." (United Nations 2003, S. 103)

Auch diese Einschätzung erscheint bei genauerer Betrachtung zu oberflächlich, da Software grundsätzlich den Charakter eines Erfahrungsgutes (vgl. hierzu Abschnitt 3.1.) aufweist, für das Monopolisierungstendenzen nicht ungewöhnlich sind (Nelson 1970). Ferner müssen auch mit der zunehmenden Portierung von unternehmenskritischen Anwendungen auf quelloffene Betriebssysteme, wie z. B. Linux, Investitionen in die Hardware getätigt werden.

Vor diesem Hintergrund sollen in diesem Beitrag drei Aspekte behandelt werden,

die in Zusammenhang mit Fragen der Standardisierung zwar immer wieder angesprochen, aber nur oberflächlich dargestellt werden:

- Kompatibilitätsprobleme in Folge von *Forking*
- Hersteller(un)abhängigkeit und
- Softwarepatente

Die Auseinandersetzung mit diesen Aspekten soll dazu beitragen, die Diskussion um OSS und damit einhergehende Fragen der Standardisierung differenzierter führen zu können.

2. Kompatibilitätsprobleme in Folge von *Forking*

Im so genannten Informationszeitalter kann kaum ein Unternehmen mehr ohne Software im Wettbewerb bestehen. Die Einsatzgebiete reichen von einfachen Schreibhilfen über komplexe Warenwirtschaftssysteme, Server für Datenbanken bis hin zur autarken Steuerung ganzer Produktionsstätten. IT-Abteilungen sehen sich mit der Anforderung konfrontiert, die bestehenden Geschäftsprozesse mit Hilfe der Informationstechnologie abzubilden und damit einen Beitrag zum Unternehmenserfolg zu leisten. Um diesem Ziel gerecht zu werden, ist es erforderlich, die bestehenden Anwendungen an die sich immer schneller ändernden Rahmenbedingungen anzupassen und die Interoperabilität bzw. Kompatibilität der verschiedenen Komponenten der EDV-Systeme zu gewährleisten.

Vor diesem Hintergrund stößt man im Zuge der Auseinandersetzung mit OSS unweigerlich auf das Problem des *Forking*s. Um dieses Problemfeld näher charakterisieren zu können, muss zunächst der Begriff der Kompatibilität präzisiert werden. Ein Großteil der Autoren, die sich mit Fragen der Standardisierung auseinandersetzen, betonen, dass dieser Begriff nur implizit und vage dargestellt wird (David und Bunn 2003, Knorr 1993, Borowicz 2001, Erhardt 2001). Grundsätzlich gelten zwei Güter als kompatibel, wenn sie in bestimmter Weise zusammenarbeiten können (Shy 2001, Martiensen 2004), d. h. Kompatibilität liegt vor, wenn es zwischen den Komponenten eines EDV-Systems möglich ist, über bestimmte Schnittstellen hinweg Informationen auszutauschen (Schmidt und Werle 1994, S. 421). Abhängig von der Beziehung dieser Komponenten untereinander lässt sich zwischen komplementärer und substitutiver Kompatibilität differenzieren; häufig wird auch von direkter und indirekter Kompatibilität gesprochen (Weiber 1992, S. 51, Hecker 1997, S. 38).

Komplementäre Kompatibilität beschreibt die Fähigkeit der Zusammenarbeit ungleichartiger Komponenten. Eine solche Beziehung besteht z. B. zwischen dem Betriebssystem (Z) und darauf aufbauender Anwendungssoftware (X), wenn beide Komponenten die Spezifikation der Schnittstelle (a) einhalten (vgl. Abbildung 1). Erfüllt gleichzeitig die Anwendung (Y) die Spezifikation der Schnittstelle (a), spielt es aus technischer Sicht keine Rolle, ob Software X oder Y verwendet wird, d. h. die Komponenten sind substitutiv kompatibel.

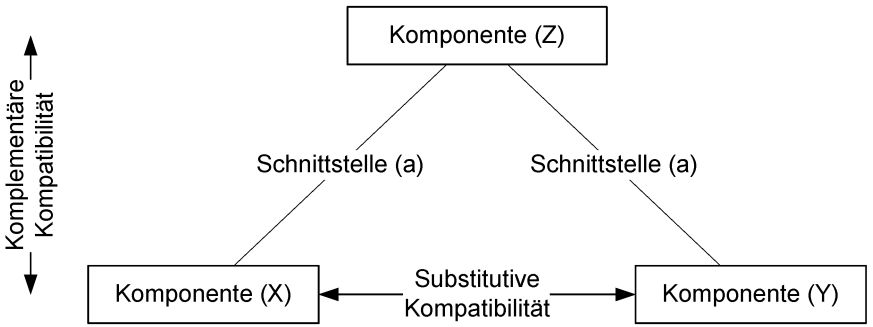


Abbildung 1: Kompatible und substitutive Kompatibilität, in Anlehnung an Pfeiffer (1989, S. 18–25) und Schmidt und Werle (1994, S. 422 f.)

Bezogen auf die Informationstechnik ist der Zweck der Standardisierung darin zu sehen, Kompatibilität zwischen den verschiedenen Komponenten eines Informationsverarbeitungssystems herzustellen (Schmidt und Werle 1994, S. 429)).

Der Begriff *Forking* umschreibt vor diesem Hintergrund die Aufspaltung eines Open-Source-Projektes in verschiedene Programmzweige, die nicht mehr uneingeschränkt kompatibel zum ursprünglichen Standard oder nur auf einen bestimmten Programmzweig abgestimmt sind. Für den Anwender bedeutet dies, dass es zu Störungen der Prozessabläufe kommen kann, mit denen hohe Folgekosten verbunden sind (hierzu auch Ricadela 2001). Zu derartigen Kompatibilitätsproblemen in Folge der Aufteilung eines Open-Source-Projektes kommt es, wenn innerhalb der Entwicklergemeinschaft unterschiedliche Ansichten bezüglich der Projektziele vorliegen (United Nations 2003, S. 102).

Grundsätzlich ist es zwar richtig, dass die Problematik des *Forkings* ein spezifisches Problem der quelloffenen Softwareentwicklung darstellt. Es wird jedoch häufig übersehen, dass sie eng mit der Wahl einer bestimmten Softwarelizenz verbunden ist. Auf diesen Zusammenhang weist beispielsweise (Rosenberg 2000, S. 109–114) hin. Die Gefahr des *Forkings* ist demnach immer dann besonders groß, wenn es aufgrund der Lizenzbestimmungen für Dritte möglich ist, das ursprünglich quelloffene Programm profitorientiert zu vermarkten. Exemplarisch sei an dieser Stelle die *Berkeley Software Distribution* (BSD) genannt, bei dieser Lizenz ist es gestattet, Weiterentwicklungen des Quellcodes als proprietäre Software zu vertreiben. Mit BSD OS, FreeBSD, OpenBSD und NetBSD existiert gleich eine Vielzahl an verschiedenen Programmzweigen des ursprünglich an der Universität Berkeley entwickelten Betriebssystems, wobei der Anreiz zum *Forking* aus kommerziellen Motiven resultiert (Rosenberg 2000, S. 111). Vor diesem Hintergrund befürchtet man eine ähnliche Entwicklung wie im Bereich der Unix-Betriebssysteme in den 80er und 90er Jahren: Hardwarehersteller wie IBM, HP oder DEC boten speziell auf ihre Hardware abgestimmte, proprietäre Unix-Distributionen an, die sich durch spezielle Zusatzfunktionen voneinander unterschieden. Hintergrund dieser Aktivitäten war das Bestreben, durch eine Differenzierungsstra-

tegie Wettbewerbsvorteile aufzubauen. Dies führte letztendlich zu einer Vielzahl von inkompatiblen Unix-Derivaten.

In den so genannten *Copyleft-Lizenzen* sieht man demgegenüber den Vorteil, dass die generelle Verfügbarkeit des Quellcodes die technische Motivation für *Forking* entfernt (Rosenberg 2000, S. 111). Grundsätzlich besteht seitens der Softwarehersteller auch Einigkeit darüber, dass der zukünftige Erfolg quelloffener Betriebssysteme maßgeblich von der Einhaltung eines Referenzstandards abhängt. So erklärte Novells Vizepräsident für das Linux-Geschäft, Jeff Hawkins: „Wir wollen uns durch Support-Angebote und einen höheren Grad an Zuverlässigkeit vom Wettbewerb unterscheiden und nicht durch das Hinzufügen inkompatibler Features" (zitiert nach Cloer und Alexander 2004, S. 4). Unternehmen wie IBM, Hewlett-Packard, Dell, Red Hat, SuSE Linux und Intel haben weiterhin angekündigt, dass sie die im August 2004 veröffentlichte Referenzspezifikationen 2.0 der Linux Standard Base (LSB) unterstützen wollen, um der Gefahr des *Forking* zu begegnen (Cloer 2004).

Die LSB ist eine Initiative der Free Standard Group, die darauf abzielt, die Kompatibilität zwischen den verschiedenen Linux-Distributionen hinsichtlich Dateisystemstruktur und den grundsätzlich notwendigen Programmbibliotheken zu vereinheitlichen, um eine Aufsplitterung in verschiedene Programmzweige zu vermeiden:

> „The LSB Specification is a binary compatibility standard. It specifies, in conjunction with other standards documents which it references, the binary environment in which an LSB compliant application executes. This is different from many computer software standards which define only application programming interfaces (APIs). Such standards apply only to building applications, but do not attempt to provide a cross–platform compatibility where each compliant application's binaries will execute without changes. Specification of the details required to ensure that applications run on each LSB-conformant platform of the same processor type means that interfaces are defined at a binary level." (Base 2004)

Aber nicht nur auf Abnehmerseite führt die Einhaltung dieses Standards zu Vorteilen. Auf Seiten der Softwarehersteller entfallen zeit- und kostenintensive Kompatibilitätstests für die Anwendungssoftware, zumal nicht mehr jede einzelne Linux–Distribution, sondern nur noch ein Standard einem diesbezüglichen Test unterzogen werden muss.[3]

3 Vor dem geschilderten Hintergrund ist auch die seit Monaten andauernde Debatte um die Öffnung des Java-Quellcodes durch Sun zu sehen. Protagonisten der Open-Source-Community, wie Eric Raymond, und Unternehmen, wie IBM, fordern die Quellcode-Freigabe der Programmiersprache, um deren Akzeptanz langfristig zu fördern, zumal es bislang für einige Linux-Distributionen nicht möglich sei, Java als Browser-Plug-In mitzuliefern. Demgegenüber sieht Sun die Gefahr verschiedener und nicht kompatibler Java-Varianten.

3. Hersteller(un)abhängigkeit

3.1. Ursachen der Herstellerabhängigkeit

Im Rahmen der Auseinandersetzung mit Standards gilt es, auf das häufig vorgebrachte Kriterium der Hersteller(un)abhängigkeit näher einzugehen. Um die Ursachen der Hersteller(un)abhängigkeit genauer verstehen zu können, gilt es zunächst,

- die Bedeutung von Netzeffekten zu erläutern und
- Software als Erfahrungsgut zu charakterisieren.

In Zusammenhang mit Netzeffekten wird häufig auch von nachfrageseitigen Skalenerträgen gesprochen, da eine linear steigende Nachfrage zu einer überproportionalen Nutzensteigerung führt (Katz und Shapiro 1986, Wiese 1990). Diese resultiert aus der Zusammenführung kompatibler Komponenten, die bei gemeinsamer Nutzung einen höheren Nutzen stiften als bei isolierter Anwendung. Das lässt sich am „klassischen" Beispiel des Telefonnetzes verdeutlichen (Katz und Shapiro 1985, S. 824). Sofern es noch nicht existiert, wird kein Konsument bereit sein, ein Telefon zu erwerben und Anschlussgebühren zu zahlen. Der Nutzen für ihn wäre gleich Null, da er mit niemandem kommunizieren könnte. Kommt es nun – aus welchen Gründen auch immer – zu einem Anstieg der Teilnehmerzahl, steigt gleichzeitig der Anreiz für potenzielle neue Teilnehmer, sich dem Netzwerk anzuschließen. Besteht das Telefonnetz beispielsweise nur aus den Teilnehmern A und B, existieren nur zwei Kommunikationswege – von A nach B und umgekehrt. Schließt sich ein weiterer Teilnehmer C dem Netzwerk an, erhält jeder der bereits partizipierenden Teilnehmer einen weiteren Kommunikationsweg; insgesamt bestehen dann sechs Verbindungsmöglichkeiten. Ein weiterer Teilnehmer D würde die Anzahl der Kommunikationswege auf zehn erhöhen usw.

Der in diesem Beispiel erläuterte Zusammenhang lässt sich allgemein durch „Metcalfe's Law" beschreiben. Es besagt, dass der Wert eines Netzwerkes – der durch die Anzahl der Kommunikationswege zum Ausdruck kommt – exponentiell mit jedem weiteren Teilnehmer wächst (Shapiro und Varian 1999). Kommt es zu einer derartigen Nutzensteigerung, spricht man von einem *direkten* Netzeffekt.

Zu *indirekten* Netzeffekten kommt es dagegen bei komplementären Komponenten. Eine solche Verbindung besteht z. B. zwischen Betriebssystem und Anwendungssoftware (Achi et al. 1995, S. 8). So stellen Softwareunternehmen aufgrund des hohen Verbreitungsgrades des Microsoft-Betriebssystems bevorzugt darauf basierende Applikationen her, um ein möglichst großes Kundensegment ansprechen zu können. Für Käufer wiederum ist die Zahl der verfügbaren Komplementärprodukte ein ausschlaggebendes Kaufkriterium für Windows, weshalb sich die Anwenderzahl wiederum erhöht und weitere Softwareunternehmen Windows-basierte Anwendungen herstellen. Erwirbt ein Konsument eine zusätzliche Einheit des Basisproduktes, steigt gleichzeitig der Anreiz für einen Komplementärguthersteller, hierauf aufbauende Anwendungen herzustellen. Durch das größere Sortiment an komplementären Produkten wird indi-

rekt der Nutzen der anderen Nutzer gesteigert. Insofern führt die starke Verbreitung eines Basisproduktes zu einem reichhaltigen Angebot an Komplementärprodukten.

Der Einfluss von Netzeffekten auf den Wettbewerb führt schließlich dazu, dass das Produkt mit dem höchsten Verbreitungsgrad überproportional bevorzugt wird. Ein Wettbewerb zwischen zwei Technologien findet deshalb nur so lange statt, bis sich der Markt für eine Technologie entschieden hat (Arthur 1989). Dabei kommt es häufig – wie im Fall des Microsoft Betriebssystems – zur Herausbildung eines (defacto-)Standards. Hierunter versteht man eine Technologie, die sich in Folge einer breiten Akzeptanz auf der Nachfrageseite im Wettbewerb durchsetzen konnte (Katz und Shapiro 1994, S. 105). In dieser Situation verfügt das Unternehmen über einen monopolistischen Preissetzungsspielraum (Hess 2000, S. 97), der zumindest vorübergehend eine überdurchschnittliche Rendite erwarten lässt. Gleichzeitig kommt es aber auch zu einem Innovationsproblem (Farrell und Saloner 1985), da neue Produkte im Vergleich zu den etablierten nur einen geringen Netznutzen aufweisen und die Abnehmer in Folge der Netzeffekte bei dem etablierten Standard verharren. Aus diesem Grund fällt es schwer, eine neue Technologie – selbst wenn diese qualitativ, z. B. hinsichtlich der Funktionalität, überlegen ist – in den Markt einzuführen.

Weiterhin gilt es, in der Auseinandersetzung mit dem Kriterium der Hersteller(un)abhängigkeit zu beachten, dass Software grundsätzlich ein Wirtschaftsgut darstellt, das durch einen hohen Komplexitätsgrad gekennzeichnet ist (Balzert 2000, S. 29 f.). In der Regel ist es deshalb für den Anwender erst nach einer gewissen Einarbeitungszeit möglich, die Software operativ zu nutzen. Insofern stellt Software ein Erfahrungsgut dar, d. h. erst nach einer gewissen Zeit ist der Anwender in der Lage, sich ein Bild über die Eignung des Produktes zu machen (Nelson 1970, S. 327). Bis dahin fallen neben den Anschaffungsausgaben Einrichtungs- und Lernkosten an, die erstere deutlich übersteigen können. Aus diesem Grund gestaltet sich der Wechsel zu einer anderen Softwarelösung häufig als problematisch. Hinzu kommt, dass die bis dahin getätigten Ausgaben irreversible Investitionen darstellen, die bei einem Produktwechsel „verloren" gehen. Nicht zuletzt aus diesen Gründen zeigen Erfahrungsgüter eine Tendenz zur Monopolbildung auf (Nelson 1970, S. 327).

Mit der Kenntnis von Netzeffekten und der Kennzeichnung von Software als ein Erfahrungsgut ist es möglich, das Kriterium der Hersteller(un)abhängigkeit genauer zu analysieren.

3.2. Open Source und Hersteller(un)abhängigkeit

Ein aktuelles Beispiel für die Auswirkungen einer Abhängigkeit im Bereich der proprietären Software ist die Beendigung der Herstellerunterstützung für Windows NT durch den Hersteller Microsoft. Die Anwender stehen vor der Wahl, auf weitere Fehlerbehebung und Support zu verzichten oder ein Update auf Windows 2000 vorzunehmen. Das Update ist nicht nur mit direkten Kosten in Form des Erwerbs neuer Lizenzen verbunden, es fallen gegebenenfalls auch noch Folgekosten an. Zum Beispiel stellt der Betrieb von Windows 2000 höhere Anforderungen an die Hardware und aufgrund der Umstellung von dem NT-Domänenkonzept auf *Windows 2000/Active Directory*

sind strukturelle Anpassungen notwendig. In Folge des hohen Verbreitungsgrades des Microsoft-Betriebssystems und der daraus resultierenden Netzeffekte kann das Unternehmen hohe Preise durchsetzen, zumal die Anwender nicht ohne Weiteres den Standard wechseln können, wenn sie auf den problemlosen Datenaustausch mit ihren Geschäftspartnern angewiesen sind. Es verwundert daher nicht, dass der Vorstandsvorsitzende von Microsoft, Steve Ballmer, den Erhalt der installierten Basis als „größte Herausforderung" ansieht (Holzwart 2004).

Der Vorteil einer quelloffenen Implementierung wird demgegenüber in der Herstellerunabhängigkeit gesehen (United Nations 2003, S. 103). Da hier grundsätzlich keine Lizenzkosten anfallen, besteht ein Kostenvorteil auf Seiten der quelloffenen Software. Allerdings stellen diese Lizenzkosten nur einen Teil der Gesamtkosten des Softwareeinsatzes dar. So fallen auch bei OSS durch kontinuierliche Fehlerbehebung, Implementierung neuer Funktionen bis hin zu neuen Designs Veränderungen an, die – wie für ein Erfahrungsgut typisch – einen entsprechenden Schulungs- und Installationsaufwand nach sich ziehen, woraus eine gewisse Produktbindung entsteht. Dabei spielt es keine Rolle, ob ein proprietärer oder offener Standard unterstützt wird. Mit der zunehmenden Portierung unternehmenskritischer Anwendungen auf Linux gilt es zudem, Kosten für leistungsfähigere Hardware zu berücksichtigen.

Der „Unterschied" zwischen den beiden Softwarekonzeptionen – bezogen auf das Kriterium der Hersteller(un)abhängigkeit – lässt sich vor dem geschilderten Hintergrund im Wesentlichen anhand von zwei Punkten verdeutlichen:

- Zeitpunkt des Updates
- Wahl eines Herstellers

Ein wichtiger Vorteil der quelloffenen Software wird darin gesehen, dass der Anwender selbst über den Zeitpunkt bestimmen kann, wann ein Update installiert oder eine Hardware-Investition getätigt werden soll (Herrmann 2004). Diese Freiheit besteht im Bereich der proprietären Software nur bedingt. Exemplarisch seien hier die restriktiv gestalteten Lizenzverträge von Microsoft genannt. Im Rahmen des so genannten „Software Assurance" Programms können Anwender zwar Updates zu vergünstigten Konditionen erwerben, allerdings sind sie mehr oder weniger dazu gezwungen, zumal Microsoft keine Updates für ältere Programmversionen zur Verfügung stellt. Mit dieser Lizenzpolitik – die auf Abnehmerseite auch auf massiven Protest gestoßen ist (Krempl 2002) – will Microsoft erreichen, dass die Anwender regelmäßig neue Softwarelizenzen erwerben und ihre Software auf dem aktuellen Stand halten.

Allerdings stellt sich in dieser Hinsicht auch die Lizenzpolitik der etablierten Linux–Distributoren, wie Red Hat und Novell/SuSE, als nicht unproblematisch dar. Um beispielsweise neue Updates der SuSE-Distribution installieren zu können, müssen die Anwender eine so genannte „Upgrade Protection" erwerben, die jährlich erneuert werden muss (Novell 2002, S. 6):

> „Customers can deploy Novell open source products by adding Upgrade Protection or Maintenance to cover their total usage. Upgrade

> Protection is a component of the Volume Licence Agreement (VLA) and Corporate License Agreement (CLA), which ensures customers always have the latest updates to their products. [. . .] Novell open source technologies require the purchase of upgrade protection for each server-/workstation where the product is installed." (Novell 2004, S. 23).

Der Erwerb einer solchen Upgrade Protection ist ferner auch Voraussetzung, um sich für das so genannte „Indemnification Program" zu qualifizieren, bei dem Anwender vor potenziellen Rechtsansprüchen Dritter – gesetzt den Fall, die Open-Source–Lösung verstößt gegen urheberrechtlich geschützten Quellcode – abgesichert werden sollen. Insofern besteht auch bei den etablierten Anbietern von Linux-Distributionen ein Zwang, kontinuierlich neue „Softwarepflegeverträge" abzuschließen. Die Freiheit, selbst über den Zeitpunkt des Softwareupdates zu entscheiden, ist deshalb lediglich im Bereich „alternativer" Linux-Distributionen, wie z. B. Debian, gegeben, die durch die Open-Source-Community entwickelt werden und bei denen derartige Pflegeverträge nicht erforderlich sind (Herrmann 2004).

Die Implementierung offener Standards führt weiterhin dazu, dass mitunter eine Vielzahl von Unternehmen in den Wettbewerb tritt. Dies begünstigt die Verhandlungsstärke der Anwender, die aus einem größeren Produktangebot wählen können. Dabei gilt es jedoch zu beachten, dass die beiden Konzepte „offener Standard" und „Open Source" nicht synonym verwendet werden dürfen, zumal auf Grundlage eines offenen Standards auch proprietäre Softwarelösungen auf dem Markt eingeführt werden, die mitunter sogar einen weitaus höheren Marktanteil als die quelloffene Software aufzeigen. Exemplarisch sei an dieser Stelle der Bereich der Applikationsserver genannt, bei dem die etablierten proprietären Produkte von IBM, BEA und Novell mit quelloffenen Softwarelösungen, wie z. B. Zope oder JBoss, konkurrieren. Bezogen auf die Hersteller(un)abhängigkeit ist es somit zu pauschal, offene Standards und Open Source als Synonyme zu betrachten.

4. Softwarepatente und offene Standards

4.1. Rechtliche Rahmenbedingungen

Die Auseinandersetzung mit offenen Standards und OSS zeigt nicht zuletzt ein Problemfeld, das sich aus Softwarepatenten ergibt, die der EU-Wettbewerbsrat gegenwärtig auf die Softwareentwicklung übertragen will (Europäische Kommission 2004). Dadurch sollen die Schutzmöglichkeiten von Software erweitert werden. So ist es beispielsweise nach dem Urheberrecht lediglich möglich, nur eine bestimmte Programmimplementierung – nicht aber das Verfahren bzw. die Idee als solche – zu schützen. Bei einem Patent lassen sich demgegenüber auch die hinter der Software stehenden Ideen schützen, wodurch der Patentinhaber ein gegenüber Dritten wirksames Ausschließlichkeitsrecht erhält. Gesetzlich wird die Patentierung von Software durch die Europäische Patentübereinkunft (EPÜ) geregelt, die bislang ein grundsätzliches Patentierungsverbot so genannter „Programme für Datenverarbeitungsanlagen" vorsieht (EPÜ Artikel 52, Absatz 2). Allerdings wurden seit Anfang der 1980er Jahre

bereits mehr als 30 000 Softwarepatente angemeldet, indem Software als „wesensbestimmender Bestandteil" eines technischen Geräts interpretiert wurde (vgl. Maaß und Scherm 2004, S. 1192). Der Zusammenhang zwischen Softwarepatenten, offenen Standards und OSS soll im Folgenden anhand des Mono-Projektes verdeutlicht werden.

4.2. Fallbeispiel: Mono

Wie eingangs bereits erläutert, spielt die Interoperabilität zwischen den verschiedenen Komponenten eines EDV-Systems eine immer wichtigere Rolle. In diesem Zusammenhang gewinnen so genannte Web Services zunehmend an Bedeutung. Darunter versteht man Programmkomponenten, die von anderen Programmen über das Internet in Anspruch genommen werden. Die einzelnen Programmkomponenten lassen sich auf der Basis standardisierter Datenformate und Internetprotokolle – wie z. B. XML oder TCP/IP – auf unterschiedliche Art und Weise miteinander kombinieren. Unter Verwendung dieser Schnittstellen können auch heterogene Systeme miteinander kommunizieren. Insofern erlauben Web Services eine flexible Modellierung und Anpassung der Geschäftsprozesse an veränderte Rahmenbedingungen (Beimborn und Weitzel 2003, S. 1360).

Gegenwärtig existieren zwei „große" Plattformen, um derartige Anwendungen zu entwickeln. Auf der einen Seite handelt es sich dabei um die *Java 2 Enterprise Edition* (J2EE). J2EE ist ein offener Standard, der auf der proprietären Programmiersprache Java von Sun beruht, deren Vorteil insbesondere in der Plattformneutralität gesehen wird, d. h. J2EE ist nicht an ein bestimmtes Betriebssystem gebunden. Auf der anderen Seite gewinnt auch das .NET-Framework von Microsoft zunehmend an Bedeutung, für die Entwicklung von webbasierten Anwendungen. Im Gegensatz zu J2EE konnte dieses Framework in der Vergangenheit allerdings ausschließlich auf Rechnern genutzt werden, die mit dem Betriebssystem Windows ausgestattet waren. Erschwerend kam hinzu, dass Microsoft Funktionen des Frameworks mit dem Betriebssystem verknüpft hat, die „traditionell" von Applikationsservern übernommen werden.

Vor diesem Hintergrund entstand das Mono-Projekt, um eine quelloffene Implementierung des .NET-Frameworks unter Linux zu realisieren. Das Interesse der Open-Source-Community begründet vor allem in der Tatsache, dass sich – im Gegensatz zu J2EE, das auf der Sprache Java basiert – unter .NET eine Vielzahl an Programmiersprachen bei der Softwareentwicklung kombinieren lässt. Die unterstützten Sprachen greifen zu diesem Zweck auf einheitliche Klassenbibliotheken und Schnittstellen zurück. Da die verschiedenen Programmiersprachen für bestimmte Probleme mehr oder weniger geeignet sind, lassen sich so die Stärken der jeweiligen Sprache im Rahmen der Softwareentwicklung kombinieren.

Die Voraussetzung für das Mono-Projekt wurde faktisch durch Microsoft selbst geschaffen, indem es die betreffenden Technologien bei der *European Computer Manufacturers Association* (ECMA) und der *International Organization for Standardization* (ISO) zur Standardisierung einreichte. Damit wurden die Funktionalitäten des .NET-Frameworks spezifiziert und die Standards sind frei zugänglich und nicht

diskriminierend. Das Problem bei der Open-Source-Implementierung ist allerdings darin zu sehen, dass die durch ECMA und ISO standardisierten Technologien nur einen Teil der .NET-Technologie abdecken. Dabei handelt es sich um die Sprache C# und die *Common Language Infrastructure* (CLI), welche die Laufzeitumgebung des .NET-Frameworks spezifizieren. Im Rahmen des Mono-Projekts werden aber auch darauf aufbauende Technologien berührt, wie z. B. ADO.NET[4] und ASP.NET[5]. Diese Technologien sind immer dann von hoher Bedeutung, wenn Anwender vollständige Kompatibilität zu Windows-Plattformen benötigen. Im Rahmen des Mono-Projekts wurden diese und andere Technologien faktisch „nachimplementiert" (Mono 2004). Verstöße gegen damit einhergehende Softwarepatente stellen allerdings ein nicht abschätzbares Risiko dar, weshalb die Erfolgsaussichten des Projektes nach wie vor umstritten sind. Mitunter wird befürchtet, dass das Entwicklungsprinzip quelloffener Software durch Patente auf Algorithmen unterlaufen und im Fall der Verletzung proprietärer Softwarepatente lizenzpflichtig werden könnte (Mueller 2001).

Im Rahmen des Mono-Projektes wird auf die patentrechtlichen Probleme auf dreierlei Weise reagiert:

> „(1) work around the patent by using a different implementation technique that retains the API, but changes the mechanism; if that is not possible, we would (2) remove the pieces of code that were covered by those patents, and also (3) find prior art that would render the patent useless." (Mono 2004)

Der Implementierung eines offenen Standards können somit patentrechtliche Probleme entgegenstehen, zumal Open-Source-Lizenzen wie die GPL die Einbindung patentrechtlich geschützter Technologien explizit untersagen. Neben den Auswirkungen auf die quelloffene Softwareentwicklung zeigen sich ähnlich gelagerte Probleme für mittelständische Software-unternehmen, die nicht über die finanziellen Mittel verfügen, um im Vorfeld umfangreiche Patentrecherchen anzustellen oder ein langwieriges Patentierungsverfahren zu bestreiten (Jakob 2003, S. 6–7). Patentrechtliche Probleme stellen insofern zwar kein spezifisches Problem quelloffener Software dar, allerdings hat diese den Nachteil, dass eine Überprüfung auf Patentverletzungen durch die Zugriffsmöglichkeit auf den Quellcode sehr viel einfacher als im Fall proprietärer Software ist.

Bei dem anhand des Mono-Projektes aufgezeigten Konflikt zwischen offenen Standards und Open-Source-Software handelt es sich nicht um einen Einzelfall. Aufsehen erregte beispielsweise der Patentstreit zwischen Kodak und Sun. Ein New Yorker Gericht bestätigte, dass Suns Programmiersprache Java Patente von Kodak verletzte (Vaske 2004). Die Unternehmen einigten sich auf einen Vergleich in Höhe von 92 Millionen US-Dollar. Gleichzeitig erklärte sich Sun dazu bereit, Lizenzen für die betreffenden Technologien zu erwerben. Es ist wichtig anzumerken, dass durch diese

4 Durch ADO.NET werden Dienste für den Datenbankzugriff bereitgestellt.

5 ASP.NET stellt die .NET-Komponente für die Entwicklung von Web Services dar und ist der Nachfolger von *Active Server Pages* (ASP).

Übereinkunft auch Lizenznehmer von Sun „geschützt" sind. Neben IBM und Bea zählen hierzu auch die Open-Source-Projekte JBoss, Geronimo und Objectweb.

5. Fazit

Fragen der Standardisierung spielen im Rahmen der quelloffenen Softwareentwicklung eine bedeutende Rolle. Bislang wurden damit einhergehende Fragen nur am Rande behandelt. Angesichts dessen ging dieser Beitrag auf drei Punkte ein, die häufig zu Missverständnissen führen bzw. nur oberflächlich dargestellt werden.

Erstens handelt es sich um das so genannte *Forking* und damit einhergehende Kompatibilitätsprobleme. Im Rahmen etablierter Open-Source-Projekte, wie z. B. im Fall von Linux, wird dieser Problematik dadurch begegnet, dass sich die führenden Softwareunternehmen auf einen Referenzstandard geeinigt haben, um die Kompatibilität der verschiedenen Programmzweige zu gewährleisten. Es bleibt abzuwarten, inwieweit diese Spezifikationen in Zukunft eingehalten und umgesetzt werden.

Zweitens wurde auf den Aspekt der Hersteller(un)abhängigkeit eingegangen. Es stellte sich heraus, dass Software – verstanden als Netzeffekt- und Erfahrungsgut – grundsätzlich eine Tendenz zur Monopolisierung aufweist. Sowohl im Bereich der quelloffenen als auch der proprietären Software besteht durch die Lizenzpolitik vieler Hersteller ein gewisser Zwang, regelmäßig neue Updates zu beziehen. Lediglich bei „alternativen" Linux-Distributionen besteht nach wie vor die Freiheit, selbst über den Zeitpunkt eines Updates zu entscheiden. Der Vorteil eines offenen Standards wurde vor diesem Hintergrund darin gesehen, dass der Anwender aus einer Vielzahl von Anbietern wählen kann, die sich selbst in einem intensiven Wettbewerb untereinander befinden. Dabei kann es sich sowohl um proprietäre als auch quelloffene Software handeln.

Drittens wurden Softwarepatente angesprochen, die die Implementierung eines offenen Standards behindern, zumal Open-Source-Lizenzen die Einbindung proprietärer bzw. patentrechtlich geschützter Technologien explizit untersagen. Allerdings stellen Softwarepatente kein spezifisches Problem quelloffener Software dar. So stellt es sich beispielsweise für mittelständische Softwareunternehmen als problematisch dar, wenn diese unwissentlich gegen ein Patent verstoßen bzw. nur über begrenzte finanzielle Mittel verfügen, um im Vorfeld umfangreiche Patentrecherchen anzustellen oder ein langwieriges Patentierungsverfahren zu bestreiten.

Wie sich die Standardisierungsbemühungen im Open-Source-Bereich in Zukunft entwickeln werden, lässt sich nicht eindeutig prognostizieren. Dies liegt nicht zuletzt daran, dass gerade in diesem Zusammenhang im politischen Umfeld widersprüchliche Entscheidungen getroffen werden. Beispielsweise positioniert sich die öffentliche Hand auf der einen Seite als Förderer der quelloffenen Softwareentwicklung. Auf der anderen Seite wird wiederum die Einführung von Softwarepatenten diskutiert, die – wie bereits dargestellt – im Gegensatz zum Entwicklungsprinzip der quelloffenen Software stehen.

Literaturverzeichnis

Achi, Z., Doman, A., Sibony, O., Sinha, J. und Witt, S. (1995), 'The paradox of fast growth tigers', *The McKinsey Quarterly* **3**, S. 4–17.

Arthur, W. B. (1989), 'Competing Technologies, increasing returns, and lock-in by historical events', *Economic Journal* **97**, S. 642–665.

Balzert, H. (2000), *Lehrbuch der Software-Technik*, 2. Aufl., Spektrum, Heidelberg, Berlin.

Base, L. S. (2004), 'Project FAQ'. http://www.linuxbase.org/modules.php?name=FAQ [19. Okt 2004].

Beimborn, D. und Weitzel, T. (2003), 'Web-Services und Service-orientierte Architekturen', *Das Wirtschaftsstudium* **32**, S. 1360–1364.

Bessen, J. (2002), 'Open Source Software: Free Provision of Complex Public Goods', http://opensource.mit.edu/ [20. Jul 2004].

Borowicz, F. (2001), *Strategien im Wettbewerb um Kompatibilitätsstandards*, Europäische Hochschulschriften, Lang, Frankfurt am Main. Reihe 5, Volks- und Betriebswirtschaft. Bd. 2802.

Chesbrough, H. (2003), 'The Era of Open Innovation', *Sloan Management Review* **44**(3), S. 35–41.

Cloer, T. (2004), 'LSB 2.0 soll Linux gegen Microsoft stärken', *Computerwoche* . http://www.computerwoche.de/index.cfm?pageid=254&artid=65075&main_id=6507 [19. Okt 2004].

Cloer, T. und Alexander, S. (2004), 'Unix-Schicksal soll Linux erspart bleiben', *Computerwoche* **31**.

Comino, S. und Manenti, F. (2003), 'Open Source vs Closed Source Software: Public Policies in the Software Market', http://opensource.mit.edu/ [20. Jul 2004].

David, P. und Bunn, J. (2003), 'The economics of Gateway Technologies and Network Evolution: Lessons from electricity supply history', *Information Economics and Policy* **3**, S. 165–202.

Demil, B. und Lecocq, X. (2003), 'Neither Market nor Hierarchy or Network: The Emerging Bazaar Governance', http://opensource.mit.edu/ [20. Jul 2004].

Erhardt, M. (2001), *Netzwerkeffekte, Standardisierung und Wettbewerbsstrategie*, Gabler / DUV, Wiesbaden.

Europäische Kommission (2004), 'Patentierbarkeit computerimplementierter Erfindungen'. http://europa.eu.int/comm/internal_market/de/indprop/comp/index.htm [10. Aug 2004].

Farrell, J. und Saloner, G. (1985), 'Standardization, Compatibility, and Innovation', *Rand Journal of Economics* **16**, S. 70–83.

Franck, E. und Jungwirth, C. (2001), 'Open versus Closed Source – Eine organisationsökonomische Betrachtung zum Wettbewerb der Betriebssysteme Windows und Linux'. http://www.isu.unizh.ch/fuehrung/Dokumente/WorkingPaper/4full.pdf.

Garzarelli, G. (2002), 'Open Source Software and the Economics of Organization'. http://opensource.mit.edu/ [20. Jul 2004].

Gfaller, H. (2004), 'Open Source und Linux – Von wegen innovativ'. ZDNet Deutschland http://www.zdnet.de/itmanager/kommentare/0,39023450,39122696,00.htm [20. Jul 2004].

Hars, A. und Ou, S. (2001), Working for Free? Motivations of Participating in Open Source Projects, *in* 'Proceedings of the 34th Hawaii International Conference on System Sciences'.

Hecker, F. (1997), Die Akzeptanz und Durchsetzung von Systemtechnologien – : Marktbearbeitung und Diffusion am Beispiel der Verkehrstelematik, Master's thesis, Universität des Saarlandes, Saarbrücken. XVII, 257 S., Diss. 1998, – Prof. Dr. Joachim Zentes.

Herrmann, W. (2004), 'Wir brauchen keinen Servicevertrag', Computerwoche, http://www.computerwoche.de/index.cfm?pageid=255&artid=65730&type=detail&kw=Wir%20brauchen%20keinen%20Servicevertrag [19. Okt 2004].

Hess, T. (2000), 'Netzeffekte: Verändern neue Informations- und Kommunikationstechnologien das klassische Marktmodell?', *Wirtschaftswissenschaftliches Studium* **29**.

Holzwart, G. (2004), 'Microsoft: Viel Cash und wenig Innovationen', Computerwoche, http://www.computerwoche.de/index.cfm?pageid=258&artid=60571&main_id=60571&category=8&currpage=1&type=detail&kw=steve%20ballmer%20installierte%20basis [22. Sep 2004].

Jakob, G. (2003), 'Positionspapier des Vereins zur Förderung Freier Software zum gegenwärtigen Stand der Diskussion um Logik- und Ideenpatente', http://ffs.or.at/Mitglieder/jack/position_swp.pdf.

Katz, M. L. und Shapiro, C. (1985), 'Network externalities, competition, and compatibility', *American Economic Review* **75**, S. 424–440.

Katz, M. L. und Shapiro, C. (1986), 'Technology Adoption in the Presence of Network Externalties', *Journal of Political Economy* **94**, S. 822–841.

Katz, M. L. und Shapiro, C. (1994), 'Systems competition and network effects', *Journal of Economic Perspectives* **8**, S. 93–115.

Knorr, H. (1993), *Ökonomische Probleme von Kompatibilitätsstandards. Eine Effizienzanalyse unter besonderer Berücksichtigung des Telekommunikationsbereichs*, Nomos, Baden-Baden.

Koch, S. und Schneider, G. (2002), 'Effort, co-operation and co-ordination in an open source software project: GNOME', *Information System Journal* **12**, S. 27–42.

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

Krempl, S. (2002), 'Wachsende Empörung über Microsofts neue Lizenzpolitik', http://www.heise.de/newsticker/meldung/28860 [22. Sep 2004].

Lakhani, K. und Wolf, R. (2003), 'Why Hackers do what they do: Understanding Motivation Effort in Free/Open Source Software Projects', http://ssrn.com/abstract=443040 [20. Jul 2004]. MIT Sloan School of Management, Working Paper 4425-03.

Lerner, J. und Tirole, J. (2000), 'The Simple Economics of Open Source', http://www.nber.org/papers/w7600 [19. Okt 2004]. National Bureau of Economic Research, Working Paper 7600.

Lerner, J. und Tirole, J. (2001), 'The Open Source Movement: Key Research Questions', *European Economic Review* **45**, S. 819–826.

Maaß, C. und Scherm, E. (2004), 'Softwarepatente', *Das Wirtschaftsstudium* **33**, S. 1192 ff.

Martiensen, J. (2004), 'Netzwerkökonomie'. Diskussionsbeitrag des Fachbereichs Wirtschaftswissenschaften der Fernuniversität Hagen; Nr. 358.

Mono (2004), *Frequently Asked Questions*, Mono-Projekt. http://www.mono-project.com/about/licensing.html [19. Okt 2004].

Morner, M. (2002), 'Das Open-Source-Software-Phänomen – organisatorisch betrachtet', *Zeitschrift Führung und Organisation* **71**, S. 219–225.

Mueller, D. (2001), 'Fängt Microsoft Open Source in seinem .NET?', ZDNet, http://www.zdnet.de/news/specials/webservice/0,39023676,2092702,00.htm [19. Okt 2004].

Nelson, P. (1970), 'Information and Consumer Behavior', *Journal of Political Economy* **78**, S. 311–329.

Novell (2002), 'Novell Lizenzvereinbarungen'.

Novell (2004), 'Novell's One Net Strategy'.

Pfeiffer, G. (1989), *Kompatibilität und Markt: Ansätze zu einer ökonomischen Theorie der Standardisierung*, Nomos, Baden-Baden.

Ricadela, A. (2001), 'The State of Software Quality', http://www.information-week.com/838/quality_poll.htm [10. Feb 2004].

Rosenberg, D. K. (2000), *Open Source – The Unauthorized White Papers*, M&T Books, Foster City.

Scacchi, W. (2002), Is Open Source Software Development Faster, Better, and Cheaper than Software Engineering?, *in* 'ICSE Workshop on Open Source Software Engineering', Orlando.

Schmidt, K. und Schnitzer, M. (2002), 'Public Subsidies for Open Source? Some Economic Policy Issues of the Software Market', http://opensource.mit.edu/ [20. Jul 2004].

Schmidt, S. und Werle, R. (1994), Die Entwicklung von Kompatibilitätsstandards in der Telekommunikation, *in* M. Tietzel (Hrsg.), 'Ökonomik der Standardisierung', Accedo, München, S. 419–448.

Shaikh, M. und Cornfold, T. (2003), 'Version Management Tools: CVS to BK in the Linux Kernel', http://opensource.mit.edu/ [20. Jul 2004].

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

Shy, O. (2001), *The Economics of Network Industries*, Cambridge University Press, Cambridge, UK.

United Nations (2003), *E-Commerce and Development Report 2003*, United Nations, New York, Genf. http://www.unctad.org/en/docs/ecdr2003ch4_en.pdf [4. Sep 2004].

Vaske, H. (2004), 'Kodak und Sun einigen sich im Patentstreit', Computerwoche, http://www.computerwoche.de/index.cfm?pageid=254&artid=65964 [19. Okt 2004].

Weiber, R. (1992), *Diffusion von Telekommunikation*, Dr. Th. Gabler Verlag, Wiesbaden.

Wiese, H. (1990), *Netzeffekte und Kompatibilität: Eine theoretische und simulationsgeleitete Analyse zur Absatzpolitik für Netzeffekt-Güter*, Poeschel Verlag, Stuttgart.

Wirtschaftsministerium (2004), 'Linux & Co. – Open Source Software in der Praxis', http://www.ebigo.de/imperia/md/content/checklisten/ebigo_compact/linux [7. Sep 2004].

von Krogh, G. (2003), 'Open-Source Software Development', *Sloan Management Review* **44**(3), S. 14–18.

Open-Source-Software als Signal

MAIK HETMANK

Dieser Beitrag befasst sich mit dem Aspekt des Signalerwerbs als Motivation zur Entwicklung von Open-Source-Software. Da Programmierkenntnisse zumeist sehr spezifisch und nicht direkt beobachtbar sind, kann es sinnvoll sein, diese Information mittelbar durch die Mitarbeit an Open-Source-Software-Projekten zu kommunizieren. Der Open-Source-Entwicklungsprozess gewährleistet die dauerhafte Sichtbarkeit sowie die Glaubwürdigkeit von solchen Signalen. Jüngere Untersuchungen zeigen empirisch, dass Signalerwerb ein wichtiges Motiv zur Mitarbeit bei Open-Source-Software darstellt. Diese Studien belegen auch, dass die Mitwirkung an Open-Source-Projekten, speziell bei sehr guten Programmierern, konkrete Auswirkungen auf das erzielte Lohnniveau haben kann.

1. Einleitung

„Fast jeder nutzt Open-Source-Produkte" lautete eine Schlagzeile bei stern.de vom 4. August 2004. Open-Source-Software (OSS) ist mittlerweile so weit verbreitet, dass nahezu jeder, und sei es nur unbewusst, auf sie zurückgreift. So nutzt Google für den Betrieb seiner Suchmaschine ein Cluster von Linux-Rechnern, und OSS-Programme wie Sendmail und Apache sind aus dem Internet praktisch nicht mehr wegzudenken. Auch im sonst eher trägen Behörden- und Verwaltungsumfeld wird zunehmend auf Open Source gesetzt. Auf starkes öffentliches Interesse stießen hier vor allem die Beschlüsse der Stadträte in Schwäbisch Hall und München, ihre Verwaltungen auf OSS umzurüsten sowie die Entscheidung der brasilianischen Regierung, künftig OSS zu bevorzugen und öffentlich finanzierte Software unter eine OSS-Lizenz zu stellen.[1]

Aber auch im Desktop-Bereich der Heimanwender gewinnt Open-Source-Software zunehmend an Bedeutung. So ringt beispielsweise der Internet-Browser *Firefox* unter Federführung der *Mozilla Foundation* dem *Internet Explorer* von Microsoft zusehends Marktanteile ab. Gerade die zunehmende Verbreitung von Open-Source–Software im Desktop-Bereich zeigt, dass sie nicht nur in Qualität, sondern auch in Bedienungskomfort proprietärer Software nahezu ebenbürtig ist. Wer aber schreibt

1 Siehe hierzu auch Fordahl (2004), Wilkens (2003*b*,*a*), Ermert und Kuri (2004), Krempl und Kuri (2004) sowie Kuri und Röttgers (2003).

solche hochwertigen OSS-Programme und vor allem warum? Schließlich ist diese Software im Allgemeinen nicht direkt monetär verwertbar. Was motiviert Entwickler und Nutzer, trotzdem an einem Projekt mitzuarbeiten, obwohl sie dafür in der Regel keine finanzielle Entschädigung erhalten und von dessen Ergebnis sie auch ohne Mitarbeit profitieren könnten? Sind sie Altruisten oder verbirgt sich dahinter doch ein rationales und ökonomisches Kalkül?

Nach Lerner und Tirole (2000) beschäftigt sich ein Entwickler nur dann mit Open–Source-Software, wenn ihm daraus ein positiver Nettonutzen entsteht. Dieser setzt sich aus aktuellem und zukünftigem Nutzen zusammen. Aktueller Nutzen umfasst direkte monetäre Entschädigungen sowie andere unmittelbare Nutzen durch die Arbeit an der Software und deren Nutzung abzüglich der Opportunitätskosten. Als zukünftige Nutzen nennen Lerner und Tirole Karriereaussichten und Reputationsgewinn. Vor allem der Aufbau von Reputation scheint, neben Eigenbedarf[2] und intrinsischen Motiven,[3] einen wesentlichen Erklärungsansatz für die Motivation zur Mitarbeit in OSS-Projekten zu liefern. OSS-Entwickler können durch den Aufbau von Reputation Signale erzeugen mit denen sie Wettbewerbsvorteile gegenüber anderen Bewerbern erzielen können. Die Kosten der Mitarbeit an OSS-Projekten lassen sich also über solche Signale in monetäre Effekte in Form von höheren Löhnen oder einer Anstellung umwandeln.

Dieser Artikel führt kurz in die Theorie des *Signallings* ein, um sich daraufhin sowohl theoretisch als auch empirisch mit dem Signalerwerb in OSS-Projekten zu befassen.

2. Zur Theorie des Signalerwerbs

Das ökonomische Problem von Informationsasymmetrien, also dem Vorhandensein von unterschiedlichen Informationsniveaus, die sich schwer auflösen lassen, wurde in seiner Bedeutung erstmals von Akerlof (1970) formuliert. Als Beispiel hat er den Gebrauchtwagenmarkt angeführt, auf dem die Verkäufer die Qualität eines Fahrzeugs zwar genau kennen mögen, sie aber den Käufern nur schwerlich glaubhaft vermitteln können, da die kommunizierten Eigenschaften nicht ohne Weiteres überprüfbar sind. Ähnlich ist es auf dem Arbeitsmarkt, wo Arbeitssuchende, deren Produktivitäten unterschiedlich, aber für Arbeitgeber nicht exakt erkennbar sind, um Arbeitsplätze konkurrieren.

Eine Maßnahme, um solche Informationsasymmetrien aufzulösen, besteht darin, private Informationen durch andere, besser beobachtbare Größen zu übermitteln (Spence 1973, 1974). Diese besser beobachtbaren Größen nennt man auch Signale. Damit sinnvolle Rückschlüsse aus den Signalen gezogen werden können, ist es jedoch entscheidend, dass diese glaubhaft und nicht ohne weiteres imitierbar sind.

Eine Separierung unterschiedlich qualifizierter Programmierer ist nur dann möglich, wenn die Kosten des Signalerwerbs sich für unterschiedlich qualifizierte Entwickler

2 Detaillierte Ausführungen zu den Motivgründen des Eigenbedarfs liefern z. B. Johnson (2001), Jäger (2002) und Hetmank (2004, S. 23 ff.).

3 Siehe hierzu auch Osterloh et al. (2004).

substantiell unterscheiden. Im Folgenden werden wir untersuchen, wie dies bei OSS–Entwicklung sicher gestellt werden kann.

3. Signalisierung durch Open-Source-Programmierung

Wenn spezifische Programmierfähigkeiten von Außenstehenden nur schwer beurteilt werden können, ist es sinnvoll, dass die Entwickler ein anderes Medium nutzen, um diese glaubhaft zu signalisieren. OSS-Programmierung kann ein Weg sein, eine solche äquivalente beobachtbare Größe zu den Programmierkenntnissen zu erzeugen. Personalverantwortliche können die spezifischen Programmierfähigkeiten dann beispielsweise anhand des Ranges der Signalgeber in der Projekthierarchie eines OSS–Projektes bewerten. Lee et al. (2003) geben hierfür ein Beispiel:

> Bewerber A hat wesentliche Module für das Linux Programm verfasst. Linux ist eine bedeutende Marke. (Dies ist der Grund, warum ich es kenne.) Da das Produkt weit bekannt ist und viel verwendet wird (besonders von Software-Experten), muss es gut sein. Es ist offensichtlich, dass Linux hohe Qualitätsansprüche hat und eine Menge guter Programmierer daran beteiligt sind. (Ansonsten wäre das Produkt nicht so erfolgreich.) *Also, wenn dieser Bewerber es in das changelog file geschafft hat, muss er sehr qualifiziert sein.* Wir sollten ihm einen entsprechenden Lohn anbieten. (S. 4, Hervorhebungen im Original, eigene Übersetzung)

Um gut sichtbare und vor allem glaubwürdige Signale zu erzeugen, ist es erforderlich, dass die Beiträge von qualifizierten Mitgliedern des Projekts angemessen und objektiv beurteilt werden. Diese Beurteilung ist jedoch nicht nur für den Außenstehenden, z. B. den Personalchef, von Bedeutung, sondern auch für den Programmierer selbst. Wenn nämlich alle Beiträge, seien sie auch noch so gering, gleich bewertet würden, sänke der Anreiz für signalsuchende Programmierer, hochwertige Beiträge zum Zwecke der Signalproduktion beizusteuern.

Die Signalanreize sind nach Lerner und Tirole (2000) umso stärker, je offensichtlicher das Leistungsniveau der entstandenen Open-Source-Software durch die relevanten Gruppen beobachtet und bewertet werden kann. Hierbei ist weiterhin von Bedeutung, wie hoch der Einfluss der persönlichen Leistung auf die Performance ist und wie viel Auskunft das Leistungsniveau der Open-Source-Software über die Fähigkeiten des Programmierers gibt.

Aufgrund der Tatsache, dass die Beiträge von mindestens gleichspezialisierten Mitgliedern des Projekts und nicht von externen oder übergeordneten Institutionen beurteilt werden, sind spezielle Bewertungsmechanismen notwendig. In vielen OSS-Projekten kommen hierfür ein Veröffentlichungsmedium sowie ein Markierungssystem für die Entwicklerbeiträge zum Einsatz. Jeder signifikante Beitrag zu einem OSS-Programm wird in Dateien oder Systemen erwähnt, in denen die Entwicklungsgeschichte der Software durch ihre zugehörigen Beiträge, Veränderungen oder Fehlerbehebun-

gen dokumentiert wird.[4] Aufgrund der Zuordnung von Beiträgen und Beitragsleistern sowie der Tatsache, dass sich die einzelnen Entwickler durch die Qualität ihrer Beiträge die Zugehörigkeit zu einer Statusgruppe erarbeiten, entsteht letztlich eine relative Leistungsbewertung der Mitglieder in einer Art Rangliste, die für Dritte beobachtbar ist.

Damit die erworbenen Signale nicht durch unzureichende und offensichtlich falsche Beurteilungen und Dokumentierungen entwertet werden, ist es notwendig, dass die Beitragsleister ausreichend hohe Drohpotentiale besitzen. Diese bestehen durch die Möglichkeit des Entzugs der Leistungen, wodurch das Projekt im schlimmsten Fall nicht weitergeführt werden könnte. Der Projektinhaber hat somit einen Anreiz, alle relevanten Beiträge durch objektives *peer review* beurteilen zu lassen (Franck und Jungwirth 2001, S. 9 f). Außenstehende können also davon ausgehen, dass die in den *history files* an entsprechender Stelle genannten Entwickler einen bestimmten Grad an Programmierfähigkeiten besitzen und die Signale aussagekräftig und glaubhaft sind.

Eine Besonderheit bei OSS-Projekten ist es, dass Programmierer gemeinsam mit anderen an der Entwicklung der Software arbeiten und die erzeugten Signale somit auch von anderen Programmierern beeinflusst werden. Für die Signalsuchenden erhöht sich durch die zunehmende Anzahl der Entwickler der Wert des Signals, da hierdurch die Attraktivität, Qualität und Größe des Projekts zunehmen. Zu viele Programmierer können jedoch auch negative Effekte für die Signalsuchenden verursachen, da der Wettbewerb um die Signale steigt. Dies kann im Extremfall dazu führen, dass die Anzahl der Entwickler größer ist als die Anzahl der zu lösenden Probleme, wodurch es zu Redundanzen kommen kann (Lee et al. 2003, S. 4). Durch die sofortige Diskussion der Lösungen, beispielsweise in Mailinglisten, wird in der Regel jedoch nur der erste Beitrag in den Programmcode eingefügt und auch nur der entsprechende Beitragsleister in der zugehörigen Dokumentation genannt.

Neue Programmierer haben also zwei Auswirkungen auf die Signalsuchenden: Zum einen steigt der Wert des Signals und damit die mögliche zukünftige Auszahlung, zum anderen sinkt jedoch die Wahrscheinlichkeit, überhaupt Reputation zu erzielen.

4. Empirische Belege für Signalproduktion durch Open-Source-Beiträge

Inwieweit Signalisierung tatsächlich für die Entwickler ein Motiv zur Mitarbeit darstellt, wurde in einigen empirischen Studien untersucht. Dabei wurde festgestellt, dass Reputation, die als Signal genutzt werden kann, für Entwickler einen wichtigen Grund zur Mitarbeit in OSS-Projekten darstellt.

Es scheint zwar auf den ersten Blick, dass die monetären Aspekte, d. h. Entlohnung und Reputation, im Vergleich zu den intrinsischen Motiven relativ selten angeführt wurden. Der Grund hierfür liegt jedoch in der Tatsache begründet, dass die Gruppe,

4 Hierzu zählen unter anderem sogenannte *history files* wie z. B. *changelog files* (siehe z. B. http://www.kernel.org/pub/linux/kernel/v2.4/ChangeLog-2.4.27) oder das *Concurrent Versions System* (CVS), eine Software zur Versionsverwaltung von Quellcode (siehe z. B. http://www.openoffice.org/issues/long_list.cgi).

Motiv	Hars und Ou	Lakhani et. al.	FLOSS-Studie Teil IV	
			bei Einstieg	bei Verbleib
Eigenbedarf	39	58	30	30
Identifikation mit der Community	28	42	19	29
Spaß	–	45	–	–
Entlohnung	14	13	4	12
Reputation	37	11	9	12
Lernen	88	41	79	71

Tabelle 1: Motive für die Mitarbeit in Open-Source-Projekten (Angaben in %), Quelle: Eigene Darstellung nach Hars und Ou (2001, S. 6 f.), Lakhani und Wolf (2005, S. 10 ff.) und Ghosh et al. (2002, S. 43 ff.)

die vorrangig extrinsische Motive hat, im Vergleich zu ihrer Bedeutung für viele Projekte unterrepräsentiert ist (Tabelle 1). In einer Untersuchung von Dempsey et al. (2002) steuerten 91 % der Entwickler nur ein bis zwei Beiträge zu OSS-Projekten bei. Lediglich 0,53 % leisteten mehr als zehn bzw. 2,2 % mehr als fünf Beiträge, trugen aber zu mehr als 6 % bzw. 13 % zum Gesamtergebnis bei (eigene Berechnungen auf Basis der Untersuchungen von Dempsey et al., S. 71).

Hars und Ou (2001) haben untersucht, inwieweit die Nennungen der Motive mit dem Engagement bzw. dem geleistetem Aufwand zusammenhängen. Hierbei zeigte sich, dass die beiden monetären Motive sowie der Eigenbedarf mit 0,3 bis 0,36 wesentlich stärker als die restlichen Motive (0,12 bis 0,19) mit dem Grad des Aufwands korrelieren (Hars und Ou 2001, S. 6 f.).

In Verbindung mit der Studie von Dempsey et al. (2002) lässt sich schließen, dass Entwickler mit wenigen Beiträgen, sogenannte Hobby- oder Freizeitprogrammierer, eher intrinsisch motiviert sind und neue Fähigkeiten erlernen möchten. Erfahrenere, qualifiziertere Entwickler, die dementsprechend auch mehr Beiträge leisten, sind hingegen eher extrinsisch motiviert. Eine streng dichotome Aufteilung in intrinsisch und extrinsisch motivierte Entwickler lässt sich jedoch nicht feststellen.[5]

Es stellt sich weiterhin die Frage, ob sich die genannten monetären Aspekte der Motivation auch tatsächlich umsetzen lassen. Einen empirischen Nachweis, dass mit der Beteiligung an OSS-Projekten reale Erträge verbunden sind, konnten Hann et al. (2004) in ihrer Studie erbringen. So fanden sie heraus, dass zwar einzelne Beiträge keinen Einfluss auf den Lohn haben, sich jedoch der Status in einem Projekt mit der Anzahl der Beiträge eines Entwicklers wesentlich erhöht. Der Status wiederum hat einen signifikanten Einfluss auf die Lohnhöhe. Diese Diskrepanz, dass ein einzelner Beitrag keinen, jedoch der Status eines Entwicklers einen signifikanten Einfluss auf die Lohnhöhe hat, erklären Hann et al. damit, dass es schwierig ist, den Zusammenhang

5 Siehe hierzu auch die Motivationsgründe untergliedert nach Programmierertypen von Lakhani und Wolf (2005, S. 10 ff.).

zwischen einem einzelnen Beitrag und dem Erfolg eines OSS-Projekts herauszufinden. Es scheint eher der Fall zu sein, dass die Menge der Beiträge, die ein einzelner Entwickler einer Statusgruppe leistet, einen wesentlichen Einfluss auf den Erfolg des Open-Source-Projekts hat (Hann et al. 2004, S. 21 f.).

Fershtman und Gandal (2004) haben in einer empirischen Studie untersucht, welche Faktoren einen Einfluss auf die Höhe der Beiträge der Entwickler haben. Sie fanden heraus, dass in Projekten, die unter einer sehr restriktiven Lizenz (z. B. der GPL) stehen, die Beiträge je Entwickler signifikant geringer sind als in Projekten mit weniger restriktiven Lizenzen (z. B. BSD-Lizenzen).[6] Sie begründen dies unter anderem mit dem sehr geringen kommerziellen Verwertungspotential der Projekte, die unter einer sehr restriktiven Lizenz stehen. Beiträge zu solchen Projekten sind vor allem aus intrinsischen oder aber Signalisierungsgründen zu erklären. Hierbei kommt es für die Beitragsleister, die an Reputation interessiert sind, darauf an, einer bestimmten Statusgruppe bzw. der Liste der „wichtigen Entwickler" anzugehören. Die Höhe der Beiträge ist für sie nicht entscheidend (Fershtman und Gandal 2004, S. 13 f.).

Die genannten Studien haben gezeigt, dass es plausibel ist, in den *Signalling*-Modellen nicht ausschließlich einzelne Beiträge als Signal für die Produktivität eines Entwicklers anzusehen, sondern diese in Verbindung mit dessen Status in einem OSS-Projekt zu betrachten. Erst wenn aufgrund der Anzahl bzw. der Höhe der Beiträge eine bestimmte Position in der Projekthierarchie erreicht wird, können die Beiträge als Signale interpretiert werden (Lee et al. 2003, Leppämäki und Mustonen 2003). Die Beitragsleister wählen dabei solche Projekte, die ihnen ein hohes Maß an Sichtbarkeit und Sicherheit ihrer Beiträge gewährleisten und in denen der Erwerb der Signale mit möglichst geringen Kosten verbunden ist.

Des Weiteren konnte mit der Studie von Hann et al. (2004) empirisch gezeigt werden, dass OSS-Beiträge, respektive die Position in einem OSS-Projekt, einen signifikanten Einfluss auf die Lohnhöhe haben und damit als Signal auf dem Arbeitsmarkt dienen können. Dabei handelt es sich nicht nur um Firmen aus dem OSS-Umfeld sondern auch um proprietäre Firmen wie Microsoft, die Entwickler aus OSS-Projekten rekrutieren (Barr 2004, Lindner 2004).

5. Fazit

Obwohl Beiträge zu Open-Source-Projekten nicht direkt monetär verwertbar sind, kann eine Mitarbeit ein rationales ökonomisches Motiv zur Signalisierung privater Informationen – den eigenen Programmierfähigkeiten – darstellen. Dieses konnte sowohl aus theoretischer Perspektive erklärt als auch aus empirischer Sicht nachgewiesen werden. Durch ein geeignetes System des *peer reviews* wird sichergestellt, dass Programmierbeiträgen ein glaubwürdiger und sichtbarer Wert zugeordnet wird. Hierdurch ist es für sie möglich, trotz der Nichtverwertbarkeit der Software reale Erträge aus ihren Beiträgen zu erzielen.

6 Die GPL wird von manchen Beobachtern als restriktiv empfunden, da sie Auflagen für die Weiterverbreitung beeinhaltet, insbesondere die Verbreitung des Quelltextes. Siehe zur ausführlichen Darstellung der unterschiedlichen Lizenzarten z. B. Grassmuck (2002, S. 275 ff.).

Signalerwerb ist natürlich nicht das einzige Motiv einer Mitarbeit an OSS-Projekten. Gerade die Vielzahl unterschiedlicher Motivationen bewirkt, dass Open Source zu einem funktionierenden und erfolgreichen System wird. Weder ausschließlich intrinsisch motivierte noch ausschließlich extrinsisch motivierte Entwickler würden den Erfolg vieler OSS-Projekte ermöglichen. Gerade die sehr restriktiven Open-Source–Lizenzen, wie etwa die GPL, ermöglichen für alle Beitragsleistenden, dass ihre Motive zur Mitarbeit zum Tragen kommen und geschützt werden.

Literaturverzeichnis

Akerlof, G. A. (1970), 'The market for lemons: Qualitative uncertainty and the market mechanism', *The Quarterly Journal of Economics* **84**, S. 488–500.

Barr, J. (2004), 'What exactly are Microsoft's plans for Linux on Windows', http://trends.newsforge.com/article.pl?sid=04/08/12/2048237.

Dempsey, B. J., Jones, D. W. P. und Greenberg, J. (2002), 'Who Is an Open Source Software Developer? Profiling a community of Linux developers', *Communications of the ACM* **45**(2), S. 67–72.

Ermert, M. und Kuri, J. (2004), 'Source: Die kommunale Welt wird bunter', http://www.heise.de/newsticker/meldung/46974.

Fershtman, C. und Gandal, N. (2004), 'The Determinants of Output per Contributor in Open Source Projects: An Empirical Examination', *CEPR Working Paper, Nr. 2650* . http://spirit.tau.ac.il/public/gandal/opensource_final.pdf.

Fordahl, M. (2004), 'Fast jeder nutzt Open-Source-Produkte', http://www.stern.de/computer-technik/computer/?id=527919&nv=hp_rt_al.

Ghosh, R. A., Glott, R., Krieger, B. und Robles, G. (2002), FLOSS Final Report - Part 4: Survey of Developers, *in* 'Free/Libre and Open Source Software: Survey and Study', International Institute of Infonomics, University of Maastricht and Berlecon Research GmbH. http://www.infonomics.nl/FLOSS/report/FLOSS_Final4.pdf.

Grassmuck, V. (2002), *Freie Software: Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn.

Hann, I.-H., Roberts, J., Slaughter, S. und Fielding, R. (2004), 'An Empirical Analysis of Economic Returns to Open Source Participation', Faculty Development Grant und Carnegie Bosch Institute, Carnegie Mellon University.

Hars, A. und Ou, S. (2001), 'Working for Free? Motivations of Participating in Open Source Projects'. Proceedings of the 34th Hawaii International Conference on System Sciences, http://csdl.computer.org/comp/proceedings/hicss/2001/0981/07/09817014.pdf.

Hetmank, M. (2004), 'Open Source Software: Eine ökonomische Betrachtung', http://maik-hetmank.gmxhome.de/Publikationen/Publikationen/Diplomarbeit_OSS-Maik_Hetmank.pdf. Diplomarbeit.

Johnson, J. P. (2001), 'Economics of Open Source Software'. http://www.polter.net/~bob/files/johnsonopensource.pdf.

Jäger, M. (2002), 'Von Mühlen und Menschen: Kleine Modellanalyse der Open Source Entwicklung nach dem Anreizprinzip', http://www.wilhelmtux.ch/files/oss.pdf.

Krempl, S. und Kuri, J. (2004), 'Münchner Stadtrat segnet Konzept zur Linux-Migration ab', http://www.heise.de/newsticker/meldung/48313.

Kuri, J. und Röttgers, J. (2003), 'Brasiliens Regierung wird Produzentin von Open Source', http://www.heise.de/newsticker/meldung/42570.

Lakhani, K. R. und Wolf, R. G. (2005), Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects, *in* J. Feller, B. Fitzgerald, S. Hissam und K. R. Lakhani (Hrsg.), 'Perspectives on Free and Open Source Software', MIT Press, Cambridge, MA. http://freesoftware.mit.edu/papers/lakhaniwolf.pdf.

Lee, S., Moisa, N. und Weiss, M. (2003), 'Open Source as a Signalling Device: An Economic Analysis'. IV. Symposium zur ökonomischen Analyse der Unternehmung, German Economic Association of Business Administration - GEABA , http://www.whu.edu/orga/geaba/Symposium/2003/B22.pdf.

Leppämäki, M. und Mustonen, M. (2003), 'Spence Revisited - Signalling with Externality: The Case of Open Source Programming'. Discussion Paper Nr. 558, http://ethesis.helsinki.fi/julkaisut/val/kansa/disc/558/spencere.pdf.

Lerner, J. und Tirole, J. (2000), 'The Simple Economics of Open Source', *Journal of Industrial Economics* **52**, S. 197–234. NBER Working Paper, Nr. 7600.

Lindner, M. (2004), 'Microsoft rekrutiert Open Source-Entwickler', http://www.pro-linux.de/news/2004/7171.html.

Osterloh, M., Rota, S. und Kuster, B. (2004), Open-Source-Softwareproduktion: Ein neues Innovationsmodell?, *in* R. A.Gehring und B. Lutterbeck (Hrsg.), 'Open Source Jahrbuch 2004: Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns, Berlin, S. 121–137.

Spence, A. M. (1973), 'Job Market Signaling', *Quarterly Journal of Economics* **87**, S. 355–374.

Spence, A. M. (1974), *Market Signaling: Informational Transfer in Hiring and Related Screening Process*, Harvard University Press, Cambridge.

Wilkens, A. (2003*a*), 'Brasiliens Präsident befürwortet Open Source', http://www.heise.de/newsticker/meldung/40129.

Wilkens, A. (2003*b*), 'Schwäbisch Hall erregt mit Umrüstung auf Linux weltweites Interesse', http://www.heise.de/newsticker/meldung/34859.

Das Microsoft-Shared-Source-Programm aus der Business-Perspektive

WALTER SEEMAYER UND JASON MATUSOW

Dieser Artikel beschreibt und begründet die strategischen Erwägungen hinter der Shared-Source-Initiative von Microsoft. Hierfür wird der Begriff des Software-Ökosystems eingeführt, der die wechselseitigen Abhängigkeiten von kommerziellen Firmen und nicht-kommerziellen Institutionen wie Universitäten beschreibt. Die Shared-Source-Initiative versucht, einige der unbestrittenen Vorteile von Open-Source-Entwicklung zu übernehmen und mit dem Geschäftsmodell von Microsoft zu vereinbaren, dass traditionell auf der Wertschöpfung durch Lizenzierung von proprietären Softwareprodukten basiert.

1. Hintergrund

In der Softwarebranche reißt die Diskussion über den Stellenwert von offener, kostenloser, kommerzieller und nichtkommerzieller Software nicht ab. Tatsächlich gibt es jedoch weitaus mehr Gemeinsamkeiten als Unterschiede. Wo aber doch Unterschiede bestehen, sollten deren Auswirkungen auf Unternehmen, Einzelpersonen, wissenschaftliche Institutionen und Regierungsbehörden näher beleuchtet werden.

In der herrschenden Diskussion um Open Source geht es um ein breites Spektrum von Themen wie Nutzungsflexibilität, Preis-Leistungs-Verhältnis, wirtschaftliche Aspekte, geistige Eigentumsrechte für Software, Industriestandards, Sicherheit, Datenschutz, Geschäfts- und Lizenzmodelle usw. Durch all diese Aspekte zieht sich einheitlich die Thematik des Zugangs zum Quellcode. Oberflächlich betrachtet ist der Quellcode ausschließlich für den Entwickler bestimmt. Dennoch stellt sich neben rein technischen Fragen der Generierung und Modifizierung des Quellcodes die grundlegende Frage nach der Zukunft von Software-Innovationen. Aus diesem Grund geht auch die Diskussion weiter.

Seltsamerweise spricht zwar fast Jeder davon, wie wichtig es ist, auf den Quellcode von Software zugreifen zu können, doch sind nur sehr wenige bereit oder in der Lage, mit diesem zu arbeiten. Um in dieser Hinsicht eindeutige und aussagekräftige Resultate zu erhalten, startete Microsoft 2002 eine Untersuchung (unveröffentlicht) bezüglich des Quellcode-Zugriffs für die in Unternehmen und Regierungsbehörden

verwendete Software.[1] Man hätte nun vielleicht erwartet, dass die IT-Experten der Unternehmen in ihrer täglichen Arbeit viel mit Quellcode umgehen. Stattdessen ergab die Untersuchung, dass ca. 95 % der Unternehmen und Behörden den Quellcode der Betriebssysteme, auf denen ihre technische Infrastruktur basiert, überhaupt nicht beachten. Darüber hinaus wurde festgestellt, dass zwar fünf Prozent der Befragten den Quellcode kennen, aber weniger als ein Prozent ihn verändert. Bei kleineren Unternehmen ist der Prozentsatz derer, die auf den Quellcode zugreifen und ihn verändern, sogar noch geringer.

Die Schwelle für das Verstehen komplexen Quellcodes ist extrem hoch. Zwar gibt es weltweit Millionen von Softwareentwicklern, doch machen diese im Vergleich zu allen Anwendern von Computern nur einen Bruchteil aus. Ferner sind die Programmierkenntnisse dieser Entwickler unterschiedlich gut ausgeprägt, so dass die Community, die hochgradig komplexe Codes handhaben kann, insgesamt noch kleiner ist. Für die meisten Unternehmen ist das Verhältnis von Kosten und Nutzen bei der Beschäftigung hochgradig qualifizierter Entwickler untragbar, vor allem weil es heute ohnehin eine Vielzahl qualitativ hochwertiger Softwarepakete gibt.[2]

Trotzdem gaben Unternehmen an, dass ihnen die Möglichkeit des Zugriffs auf den Quellcode von Betriebssystemen wichtig sei.[3] Die Mehrzahl der Unternehmen und Regierungsbehörden befürworteten die Option eines zugänglichen Quellcodes. Einfach ausgedrückt: Transparenz erhöht das Vertrauen.

Das legt den Schluss nahe, dass es für die meisten Anwender weit wichtiger ist, die Möglichkeit zu haben, etwas zu tun, als es dann tatsächlich auch zu tun. Ein gutes Beispiel hierfür ist die gesetzlich vorgeschriebene Offenlegung der Bilanzen von Aktiengesellschaften. Zwar sind diese Bilanzen öffentlich zugänglich, doch sind sie äußerst kompliziert. Um den Status einer Gesellschaft bewerten zu können, ist ein fundiertes Finanzwissen erforderlich. Der Großteil der privaten Investoren benötigt daher die Hilfe einer kleinen Gruppe von Fachleuten, die für sie die Zahlen deuten und Empfehlungen aussprechen. Die Möglichkeit, die Zahlen einzusehen, ist also für alle gegeben: Aufgrund der gegebenen Transparenz besteht daher Vertrauen. Die meisten Investoren werden jedoch von dieser Möglichkeit nie Gebrauch machen.

Der private Investor und der typische Anwender heutiger Betriebssysteme befinden sich in einer ähnlichen Situation. Die meisten Unternehmen oder Einzelpersonen haben gar nicht vor, in die Tiefen des Betriebssystems vorzudringen und den Quellcode zu verändern.[4] Unternehmen und der durchschnittliche Konsument hängen stark von

1 An diesen privaten Forschungsarbeiten waren mehr als 1110 Personen aus fünf Ländern beteiligt, darunter Entscheidungsträger von Unternehmen, IT-Fachleute und Entwickler. Die Studie wurde im April 2003 abgeschlossen. Befragt wurden Personen welche mit Open-Source-Software (OSS) und/oder Microsoft-Systemen arbeiten und die Möglichkeit haben, auf Quellcode zuzugreifen.

2 Diese Annahme basiert auf der stark vereinfachten Auslegung des Transaktionskosten-Konzepts von Ronald Coase und dessen Einfluss auf das Verhalten von Unternehmen. Coase erhielt 1991 den Nobelpreis für Wirtschaftswissenschaften für seine Arbeiten in Bezug auf die Bedeutung von Transaktionskosten und Eigentumsrechten für die institutionelle Struktur und Funktion der Wirtschaft (http://coase.org/, Stand 20. Mai 2003).

3 Die in Fußnote 1 genannte Studie ergab, dass ca. 60 % der Befragten die Möglichkeit zum Einblick in den Quellcode für die Nutzung von Software in einer Geschäftsumgebung für unerlässlich halten.

4 Dies gilt in gleichem Maße für Windows, Linux, Mac OS, Netware, OS/400 und andere bedeutende

kommerziellen Anbietern ab, die ihnen das erwartete Qualitäts- und Supportniveau garantieren. Und genau an dieser Stelle liefern kommerzielle Softwareanbieter einen Mehrwert für die von ihnen entwickelten und vertriebenen Produkte.[5]

Vor drei Jahren wurde die Shared-Source-Initiative bei Microsoft ins Leben gerufen. Im Rahmen dieser Initiative stellen wir verschiedene Arten von Microsoft-Quellcode für Kunden, Regierungsbehörden, Partner und Mitbewerber in der ganzen Welt bereit. Ein Teil des Quellcodes – wie der von Windows – wird nur zur Einsichtnahme verfügbar gemacht (d. h. es können keine Änderungen durchgeführt werden), während andere Programme, die Technologien wie Windows CE.NET betreffen, die Änderung und den Weitervertrieb des Quellcodes erlauben.[6]

Im Zuge unserer Tätigkeit in der Shared-Source-Initiative haben wir unzählige Gespräche mit Einzelanwendern und Unternehmen über die Bedeutung von Quellcode für ihre speziellen Anforderungen geführt. Obwohl wir mehr als 800 000 Entwicklern den Quellcode an bekannt gegeben haben, ist das nur ein sehr geringer Prozentsatz aller Entwickler, die mit Microsoft-Technologien arbeiten. Unsere praktischen Erfahrungen auf globaler Ebene bestätigen, dass die operationalen Bedürfnisse, wie oben erwähnt, mit dem Beruhigungsfaktor der Transparenz zusammenhängen. Bei dem Ansatz, unseren Quellcode zu lizenzieren, spielen ebenfalls Faktoren wie Transparenz, Wahlmöglichkeiten, Vertrauen und Bedürfnisse eine wichtige Rolle.

Unser Ansatz beruht auf drei einfachen Grundgedanken: Erstens verlangen die Kunden Zugriff auf den Quellcode aus technischen Gründen und aufgrund der Tatsache, dass Transparenz das Vertrauen steigert. Zweitens gibt es keinen einheitlichen Weg der Bereitstellung von Quellcode durch Microsoft, um alle geschäftlichen und lizenztechnischen Anforderungen in der gesamten Produktpalette zu erfüllen. Drittens ist der Kunde beim Umgang mit dem Quellcode erfolgreicher, wenn er zusätzlich zur Technologie solide Tools und Informationen erhält. Bei der Entwicklung des Shared–Source-Ansatzes hat Microsoft diese Grundgedanken berücksichtigt. Shared Source ist nicht gleichbedeutend mit Open Source. Vielmehr ist es ein Mittel für Unternehmen, die Software direkt vermarkten, um Quellcode bereitzustellen, ohne dabei ihre Alleinstellungsmerkmale und ihr Geschäftsmodell zu schwächen. Microsoft erkennt die Vorteile des Open-Source-Modells zwar an, doch gilt es zu berücksichtigen, dass dieses Modell nicht universell anwendbar ist.

Diese Abhandlung verfolgt zwei Zielsetzungen: Erstens sollen die Shared-Source–

---

kommerzielle Betriebssysteme. Schon das kleinste von ihnen umfasst Millionen von Zeilen an Quellcode (ein Betriebssystem ist mehr als nur ein Kernel), wobei die Komplexität dieser Systeme im Verlauf ihrer Entwicklung eher zunimmt als abnimmt.

5 Natürlich dürfen die Fähigkeiten eines Community-Supports nicht unterschätzt werden. Seit Jahren bestehen Newsgroups und Mailinglisten, in denen die Community Support für kommerzielle, offene, kostenlose und als Shareware verfügbare Software leistet. Die Unternehmen bevorzugen im Hinblick auf ihre unternehmenskritischen Systeme allerdings eher einen professionellen Support mit Service Level Agreements zur Eindämmung der Risiken.

6 Mit Stand vom Mai 2003 bietet Microsoft entsprechende Programme für Windows, Windows CE.NET, Visual Studio .NET, C#/CLI, ASP.NET und Passport an. Für die Windows-Programme werden nur akademisch tätigen Wissenschaftlern Rechte zur Veränderung des Quellcodes gewährt. Für alle anderen Quellcode-Programme werden Rechte zur Änderung und zum Weitervertrieb eingeräumt. Weitere Informationen finden sich unter http://www.microsoft.com/sharedsource/.

Initiative sowie kommerzielle Software in den breiteren Kontext der derzeitigen Debatte über die Quellcode-Lizenzierung gestellt werden. Zweitens soll erläutert werden, wie Microsoft die Lizenzierung seiner zentralen Ressourcen an geistigem Eigentum angeht.

2. Allmähliche Annäherung

2000 und 2001 schienen die Fronten der Beteiligten der Quellcode-Debatte klar gezogen: Microsoft galt als eindeutiger Pol in der ständig neue Positionen hervorbringenden Debatte mit den traditionellen Verfechtern geistigen Eigentums auf der einen Seite und den Gegnern kommerziellen Eigentums an einer Software auf der anderen Seite. Dem Open-Source-Ansatz verschriebene Einzelpersonen und Unternehmen stellten sich bewusst als Alternative zu Microsofts Geschäftspraktiken oder gar als aktive Gegner von Microsoft dar. Heute, 2003, hat sich infolge der Nachwirkungen des Dot.com-Zeitalters ein pragmatischer Ansatz bei Unternehmen wie auch bei Einzelpersonen durchgesetzt.

Open-Source-Software (OSS) ist als Softwarekategorie in kommerzielle und nichtkommerzielle Sektoren unterteilt. Für viele finden die interessantesten OSS-Fortschritte heute in der kommerziellen Kategorie statt, da bedeutende Investitionen, Ressourcen und Technologien von Seiten all derer einfließen, die OSS als Grundlage für strategische Geschäftszwecke nutzen wollen.

Ein genauer Blick auf die kommerzielle Software-Community zeigt, dass sich die Quellcode-Lizenzierung bei den wichtigsten Anbietern durchgesetzt hat. Auf dem heutigen Markt bilden Softwareentwicklung, Lizenzierung und Geschäftsstrategien eine Mischung aus Community-basierten und kommerziellen Modellen. Nur wenige Softwareunternehmen können sich heute noch ausschließlich als OSS-orientiert (im Sinne von OSS als durch die Community entwickelte und nicht gewinnorientierte Software) oder rein kommerziell bezeichnen.

Für diese Diskussion soll zunächst einmal eine klare Linie zwischen nichtkommerzieller und kommerzieller Software gezogen werden. Die wichtigen Rollen beider Kategorien im Rahmen des in den letzten 30 Jahren entstandenen Software-Ökosystems werden später in dieser Abhandlung erörtert.

Nichtkommerzielle Software lässt sich grob in drei Kategorien aufteilen:

Forschung Regierungsbehörden und akademische Wissenschaftler entwickeln Technologien, die den allgemeinen Entwicklungsstand vorantreiben sollen.

Lehre Professoren, Studenten und Autodidakten arbeiten und lernen mit kostenloser Software, wobei ihre Forschungsergebnisse nicht der kommerziellen Vermarktung dienen.

Entwicklung und Problembehandlung innerhalb der Community Freizeitprogrammierer oder professionelle Entwickler erstellen Software ohne kommerzielle Absichten, so z. B. Software, die bestehende kommerzielle Programme ersetzt oder Probleme behebt, die vom Hersteller nicht berücksichtigt wurden.

Kommerzielle Software lässt sich grob in zwei Kategorien aufteilen:

Direkte Vermarktung Von der Community und/oder von Unternehmen entwickelte Produkte werden zur Generierung direkter Einnahmen verwendet.

Indirekte Vermarktung Von der Community und/oder von Unternehmen entwickelte Produkte werden für die gewinnbringende Vermarktung anderer Produkte oder Dienstleistungen eingesetzt.

Hierbei ist anzumerken, dass die Konzepte der kommerziellen und nichtkommerziellen Software nichts mit der Zugänglichkeit des Quellcode zu tun haben. Wenn der langjährige Anbieter einer kommerziellen Software den Quellcode eines bestimmten Produkts bekannt gibt, ändert das nichts an dem kommerziellen Ursprung der Software. Gleichzeitig kann ein kommerziell auftretendes Unternehmen eine gemeinschaftliche, nichtkommerzielle Software nutzen, ohne dass diese Software dadurch ihren Status als nichtkommerzielle Software einbüßt.

In der folgenden Liste werden die oben angeführten kommerziellen Kategorien anhand von Beispielen aus der Softwarebranche dargestellt. Obwohl sich viele der unten aufgeführten Unternehmen der OSS-Bewegung zuordnen, sind sie doch eindeutig kommerzielle Unternehmen. Einige der genannten Unternehmen verfolgen wiederum das OSS-Konzept nicht in direkter Weise und können doch aufgrund ihres Ansatzes bei der Softwarevermarktung als Beispiel dafür herangezogen werden.

Direkte Vermarktung von Community-Entwicklungen Die Linux-Versionen von Red Hat, Inc. und SuSE sind Kombinationen aus von der OSS-Community entwickelter Software und vom Unternehmen finanzierten, professionellen Entwicklungsbeiträgen. Die Preisgestaltung der *Premium Editions*, die Zertifizierungsverfahren für Hardware und Anwendungen sowie die Support-Modelle von Red Hat dienen alle als Mechanismen zur direkten Vermarktung des Betriebssystems.

Apple Computer, Inc. hat Community-Software mit kommerzieller Software kombiniert, um das Betriebssystem OS X zu erstellen. Das Unternehmen vermarktet die Software direkt und verwendet von der Community entwickelten Code.

Direkte Vermarktung von Unternehmens-Entwicklungen Microsoft hat das Windows-Produkt unter Nutzung unternehmenseigener Entwicklungsressourcen erstellt. Das Produkt wird durch die Lizenzierung seiner Binärversion direkt vermarktet. Der Quellcode ist nun im Rahmen der Shared-Source-Initiative für eine begrenzte Community verfügbar.

CollabNet, Inc. hat ein proprietäres Tool entwickelt, das durch eine Lizenzierung seiner Binärversion sowie durch zugehörige Dienste direkt vermarktet wird. Das Produkt erleichtert die Nutzung des OSS-Entwicklungsmodells, das zur Entwicklung nichtkommerzieller Software eingesetzt werden kann.

Indirekte Vermarktung von Community-Entwicklungen IBM Corp. spielte eine wichtige Rolle in der Community-basierten Entwicklung des Apache-Webservers. Obwohl IBM den Apache-Server nicht direkt vermarktet, generiert das Unternehmen durch den Verkauf des WebSphere-Produktes Einnahmen aus dieser Investition.

RealNetworks, Inc. veröffentlichte entscheidende Segmente des Quellcodes seines Produktes Helix, das ursprünglich kommerziell entwickelt wurde. Ziel der Community-Entwicklung im Zusammenhang mit der Helix-Produktpalette ist die Erschließung eines breiteren Marktes für andere wertschöpfende Produkte.

Indirekte Vermarktung von Unternehmens-Entwicklungen Der Acrobat Reader von Adobe Systems, Inc. ist eine Verbindung aus unternehmenseigener Entwicklung und streng gewahrtem geistigen Eigentum.[7] Der Reader steht kostenlos zum Download bereit, um so den Verkauf der Vollversion des Acrobat-Produktes zu fördern.

Driver Development Kits (DDKs) und *Software Development Kits* (SDKs) werden von allen Anbietern kommerzieller Betriebssysteme bereitgestellt (so z. B. das NDK von Novell, Inc. und das DDK von Microsoft). Diese Entwicklertools enthalten häufig Beispiele für Quellcode, die verändert und weiterverbreitet werden können, obgleich die Kits den Entwicklern kostenlos zur Verfügung gestellt werden. Die DDKs oder SDKs selbst haben keinen direkten kommerziellen Wert, doch schaffen sie für andere die Gelegenheit, Software und Hardware für die jeweilige Plattform zu erstellen.

Oftmals wird fälschlicherweise angenommen, dass nach dem Open-Source-Modell entwickelte Software zufällig von einer Gruppe verstreut arbeitender Entwickler geschrieben und dann von Unternehmen übernommen wird. Das mag zwar für kleinere Projekte zutreffen, doch werden die zentralen OSS-Technologien von professionellen Teams in Unternehmen oder hochgradig strukturierten gemeinnützigen Organisationen erstellt, getestet, vertrieben und gewartet. Das hinter dem Zugriff auf Quellcode stehende Konzept hat mit der Unterscheidung, ob eine Software kommerziell ist oder nicht, nichts zu tun. Der Zugriff auf den Quellcode ist sowohl im kommerziellen als auch nichtkommerziellen Umfeld von Bedeutung.

Die Problematik des Quellcodes hat zweifellos Einfluss auf die Innovationen in der Branche. Die oben erwähnte Annäherung ist das Ergebnis dieses Einflusses auf die heutige Branche.

3. Das Software-Ökosystem

Grundlage jeglicher Softwareentwicklung ist die Interaktion zwischen von Regierungsbehörden, akademischen Institutionen und privaten Stellen betriebenen Forschungsbemühungen. Diese Beziehungen stellen ein verknüpftes, natürliches Ökosystem dar.

7 Adobe Systems, Inc. stellt zwar die Spezifikation des Dateiformats PDF zur Verfügung, legt jedoch nicht den Quellcode für die Implementierung offen. Weitere Informationen unter http://www.adobe.com.

Obwohl diese Organe unabhängig voneinander bestehen und unabhängige Entwicklungen vorantreiben, bestehen ganz klar Abhängigkeiten, durch die erheblich weitreichendere Ergebnisse für die Allgemeinheit erbracht werden.

Dieses Ökosystem ist der Motor für fortwährende Innovationen, die den Bereich der Informationstechnik zu einer der dynamischsten Wirtschaftsbranchen gemacht haben.[8] Die Zusammenführung verschiedener Entwicklungs-, Lizenz- und Geschäftsmodelle war dabei der entscheidende Faktor für den Erfolg.

Grundlagenforschung wird von Regierungsstellen und Universitäten erbracht und deren Ergebnisse werden der Allgemeinheit zur Verfügung gestellt.[9] Einige der so entstehenden Technologien werden wiederum im privaten Sektor aufgegriffen und mit stetig steigenden Forschungs- und Entwicklungsetats[10] weiterverfolgt, um neue kommerzielle Produkte zu entwickeln und gleichzeitig die Arbeit von Standardisierungsgremien zu unterstützen. Die Erfolge im privaten Sektor führen zu mehr Beschäftigung, mehr Steueraufkommen und einer zusätzlichen Finanzierung akademischer Forschungsprojekte.[11]

Die mit dem Software-Ökosystem verbundenen Konzepte gelten natürlich nicht nur für die Diskussionen über den Zugriff auf den Quellcode. Ein Beispiel hierfür ist die Luftfahrt. Obgleich die meisten Personen in ihrem Alltag eher wenig mit einem F-15 Eagle-Kampfjet anfangen können, hat die zivile Luftfahrt dennoch von den staatlichen und akademischen Forschungen in diesem Bereich profitiert (angefangen von der Metallurgie bis hin zu den Anzeigedisplays im Flugzeug).

Übertragen auf die IT-Branche könnte man das Beispiel TCP/IP heranziehen. Diese

8 Allein in der zweiten Hälfte der 1990er Jahre machten datenbezogene Sektoren 8,3 % der gesamten US-Wirtschaft aus, wobei sie für ca. 30 % des gesamten Wirtschaftswachstums sowie mindestens für die Hälfte des Anstiegs der Produktivitätsraten verantwortlich waren (U.S. Department of Commerce, U.S. Government Working Group on Electronic Commerce, „Leadership for the New Millennium: Delivering on Digital Progress and Prosperity", 16. Januar 2001).

9 In der Tat sind US-amerikanische Bundesbehörden gesetzlich verpflichtet, die Empfänger von Zuschüssen sowie Vertragsnehmer der öffentlichen Hand zu einer Patentierung der Ergebnisse von staatlich geförderten Forschungsarbeiten anzuhalten, und die Universitäten bemühen sich aktiv um die Sicherung von Urheberrechten im Zusammenhang mit ihren Forschungsarbeiten. Mindestens seit 1980 verfolgt die Regierung der USA eine strikte Politik der Bereitstellung von Ergebnissen aus von ihr geförderten technologischen Forschungen an die Industrie, um so deren kommerzielle Nutzung sowie Innovationen zu fördern. Siehe hierzu den Bayh-Dole Act von 1980, den Stevenson-Wydler Technology Innovation Act von 1980, den Federal Technology Transfer Act von 1986, Executive Order 12591 von 1987, den National Technology Transfer and Advancement Act von 1995 und den Technology Transfer Commercialization Act von 2000.

10 Von 1969 bis 1994 beliefen sich die US-amerikanischen Investitionen in Forschung und Entwicklung im Hochtechnologiebereich auf 77,6 Milliarden US-Dollar seitens der Regierung und 262 Milliarden US-Dollar seitens der privaten Industrie (Industrial Research and Development Information System (IRIS) der National Science Foundation, Stand vom 11. Oktober 2002 unter http://www.nsf.gov/sbe/srs/iris/start.htm).

11 Ein gutes Beispiel hierfür ist Google, Inc. Google war als eines von 15 Phase-1-Projekten der Universität Stanford im Rahmen der *Digital Libraries Initiative* staatlich gefördert worden. 1996 wurde die Technologie gegenüber dem Office of Technology Licensing (OTL) der Universität Stanford offen gelegt. 1998 erteilte das OTL Sergey Brin und Larry Page die Erlaubnis zur Gründung einer kommerziellen Struktur auf Basis der Technologie. Heute ist Google, Inc. ein erfolgreiches Unternehmen, das Einkünfte sowohl für sich als auch für die Universität Stanford generiert.

Technologie ging aus einem staatlichen Forschungsprojekt hervor, das in den Universitäten nach dem OSS-Modell weiterentwickelt und schließlich zu einem offenen Industriestandard wurde. Seitdem wurde sie laufend verbessert und hielt durch proprietäre Implementierungen kommerzieller Softwareunternehmen wie Novell, Apple, IBM und Microsoft in den meisten Rechnern Einzug.

Das Betriebssystem Windows wurde hingegen von Microsoft selbstständig und gewinnorientiert entwickelt. Das Produkt verwendet dennoch viele Komponenten, die aus akademisch oder staatlich geförderten Projekten entstanden sind, sowie Dutzende offener Industriestandards. Ferner bot die Veröffentlichung unzähliger Schnittstellen für die Anwendungsentwicklung Geschäftsmöglichkeiten für eine Vielzahl von Softwareunternehmen, aus denen wiederum auf ganz bestimmte Anforderungen ausgelegte, spezifische Anwendungen entstanden.

Was bedeutet dies nun also für die Zukunft? Wenn man von der Vergangenheit ausgeht, wird die Zukunft der Software nicht das Ergebnis eines einzelnen Entwicklungs-, Lizenz- oder Geschäftsmodells sein. Zukünftige Innovationen werden nicht allein durch den Staat, die Privatindustrie oder lockere Zusammenschlüsse von Einzelpersonen erreicht werden, die zum Wohl der Gesellschaft arbeiten. Ob der Prozess der ständigen Innovationen, wie er bereits über 30 Jahre anhält, auch weiter erfolgreich sein kann, hängt einzig und allein davon ab, ob auch weiterhin Ansätze und Technologien miteinander verschmelzen können. Letztendlich werden sowohl Konsumenten von kundenspezifischen Programmen wie auch Kunden von Paketlösungen die Nutznießer dieser Entwicklung sein, da es letztlich die Kräfte des Marktes sind, die alle Aktivitäten und Ergebnisse von Unternehmen und Einzelpersonen bestimmen.

4. Das richtige Verhältnis

Angesichts der zunehmenden gegenseitigen Annäherung der Softwareanbieter und der Auswirkungen des Software-Ökosystems stellt sich für Microsoft nun die Herausforderung, ein ausgewogenes Verhältnis von mehr Transparenz, nachhaltigen Geschäftsmöglichkeiten und Investitionen in Innovationen zu finden.

OSS hat sicherlich Auswirkungen auf die Einstellung der Softwareunternehmen zur Frage des geistigen Eigentums und dessen Handhabung.[12] Neben den Vorteilen, die ein gemeinsam genutzter Quellcode beinhalten kann, stellt er auch eine Herausforderung für die gängigen Konzepte der Vermarktung von Software und ihrer Abgrenzung im Wettbewerb dar. Die Vorgehensweise der meisten Softwareunternehmen bestand bislang in einem strikten Schutz der geistigen Eigentumsrechte an ihren Softwareprodukten, um deren Alleinstellungsmerkmale und Wettbewerbsfähigkeit am Markt zu schützen. Die Gesetze in Bezug auf Geschäftsgeheimnisse, die viele Aspekte der Software abdecken, die durch die Kompilierung verborgen sind, spielen eine zentrale Rolle bei der Strategie zum Schutz geistigen Eigentums der meisten kommerziellen Softwareunternehmen. Früher wurde dieser Schutz entweder durch den Vertrieb

12 Im Rahmen der Diskussion um die Lizenzierung von Quellcode wird häufig auch über die Verwendung des Ausdrucks „geistiges Eigentum" gestritten. Hier umfasst dieser Ausdruck das gesamte Konzept von Urheberrechten, Patenten, Geschäftsgeheimnissen und Warenzeichen.

reiner Binärversionen oder eine Quellcode-Weitergabe im Rahmen von Geheimhaltungsvereinbarungen sichergestellt. Heute müssen Unternehmen jedoch aufgrund der OSS und anderer Source-Sharing-Modelle ein ausgewogenes Verhältnis zwischen dem Schutz der geistigen Eigentumsrechte (insbesondere in Bezug auf den Schutz von Geschäftsgeheimnissen) und dem Nutzen des Kunden/Partners finden.

Durch den Wunsch nach mehr Transparenz sind Urheberrechte in den Vordergrund gerückt. Die kreative Kombination und Anwendung aller vier Optionen zum Schutz geistigen Eigentums ebnet den Weg für einen verstärkten Austausch von Quellcode, wobei die Unternehmen ihre Geschäftsgeheimnisse nun selektiv schützen können.[13]

Das heißt nicht, dass es in jedem Fall für ein Unternehmen angebracht ist, den Quellcode zu seinen Produkten preiszugeben und damit seine Geschäftsgeheimnisse ganz oder teilweise zu offenbaren. Kommerzielle Softwareunternehmen wägen Kundenanforderungen und -wünsche gegen eine Reihe anderer Geschäftsfaktoren ab. So fordern beispielsweise viele Investoren, dass Unternehmen ihr Vermögen mit allen Mitteln schützen sollen, um zukünftige Anlagenrenditen und stabile Einnahmen zu sichern. Das wird jeder bestätigen können, der einmal versucht hat, Kapital für ein Softwareunternehmen aufzubringen. Mitunter kann es aber durchaus sein, dass das Geschäftsgeheimnis eines Softwareproduktes tatsächlich seinen Marktvorteil ausmacht. In diesem Fall wäre das betreffende Unternehmen natürlich nicht bereit, den Quellcode hierfür preiszugeben.

Die erfolgreichsten Softwareunternehmen investieren erhebliche Summen ihrer Bruttoerträge in Forschung und Entwicklung. Microsoft investiert derzeit ca. 5 Milliarden US-Dollar jährlich, bzw. ca. 15 % des Bruttoertrags, in die Zukunft des Unternehmens.[14] Aber wohin führt dieser Weg? Wie ist ein ausgewogenes Verhältnis zwischen den offenkundigen Vorteilen der Quellcode-Transparenz und Flexibilität für Entwickler und den Realitäten der Softwarebranche zu erreichen, die ihre Ressourcen schützen und sichere Einnahmequellen gewährleisten muss?

Jedes Unternehmen muss hier seinen eigenen Weg finden. Für Microsoft stand fest, dass Kunden, Partner und Regierungsstellen mehr Transparenz und Flexibilität wünschten. Andererseits mussten wir auch die andere Seite der Gleichung im Auge behalten. Daraus entstand die Shared-Source-Initiative.

5. Die Shared-Source-Initiative

Microsoft stellt Kunden, Partnern und Regierungsbehörden weltweit Quellcode zur Verfügung. Wir haben Quellcode-Programme auf den Markt gebracht, die weit über 100 Millionen Zeilen Quellcode liefern. Die Shared-Source-Initiative entwickelte sich aus dem Versuch, sowohl auf die Forderungen unserer Kunden und Partner nach mehr Quellcode-Zugriff einzugehen als auch die Vor- und Nachteile des OSS- und Free-Software-Ansatzes sorgfältig zu evaluieren. Schließlich haben wir gezielt unsere

13 Die in jüngster Zeit aufgekommene Diskussion um das Thema Patente und Linux hat gezeigt, wie wichtig Klarheit in diesem Bereich ist.

14 Viele Softwareunternehmen investieren 15 bis 30 Prozent ihres Bruttoertrags in Forschung und Entwicklung.

Erfahrungen mit diesen beiden Ansätzen sowie unser bestehendes Geschäftsmodell angewandt, um besser auf die Kundenbedürfnisse eingehen zu können.

Shared Source ist ein Framework und keine Lizenz.[15] Jedes kommerzielle Softwareunternehmen muss das Zusammenspiel aus Entwicklungsmodellen, Lizenzierung und Geschäftsmodellen auswerten, um eine erfolgreiche Strategie aufstellen zu können, bei der Quellcode gemeinsam genutzt oder veröffentlicht werden kann, wobei sowohl die Vorteile für die Kunden als auch die Wahrung der Wettbewerbsfähigkeit des Unternehmens berücksichtigt werden. Das Lizenzierungs-Konzept von Microsoft reicht daher von Rechten zur bloßen Einsichtnahme (wobei der Lizenznehmer den Quellcode von Microsoft zu Referenzzwecken und zum Debugging einsehen kann, Änderungen oder ein Weitervertrieb jedoch ausgeschlossen sind) bis hin zu weitreichenden Rechten, die dem Lizenznehmer die Prüfung und Änderung sowie den Weitervertrieb und Verkauf abgeleiteter Programme ohne Lizenzgebühren an Microsoft erlauben.

Inzwischen besitzen Hunderttausende von Entwicklern Quellcode von Microsoft. Wir haben damit das wohl kommerziell wertvollste geistige Eigentum der Softwarebranche für Tausende von Unternehmen in mehr als 60 Ländern zugänglich gemacht.[16] Shared-Source-Programme bieten nun Quellcode für Windows, Windows CE .NET, Visual Studio .NET, C#/CLI, ASP.NET und Passport-Technologien. Wir werden auch weiterhin den Wert des Quellcodes als wertvolle Eigenschaft unserer Produkte sowie die Möglichkeit zur Nutzung durch unsere Kunden und Partner abwägen.

Eines der größten Missverständnisse bezüglich des Shared-Source-Modells ist die Annahme, dass es auf eine einzelne Lizenz und die Ansicht des Quellcodes ohne Möglichkeit zur Änderung beschränkt sei. Tatsächlich umfasst Shared Source vier zentrale Konzepte:

Support für bestehende Kunden Bestehenden Kunden wird Quellcode-Zugriff gewährt, um den Produkt-Support, die Implementierung, Sicherheitstests und kundenspezifische Anwendungsentwicklungen zu erleichtern.

Entwicklung neuer Programme Anhand von Beispielen und Kernkomponenten wird Quellcode zu Übungszwecken bereitgestellt, um neue Entwicklungsprojekte voranzutreiben.

15 Die *Open Source Initiative* hat 43 Lizenzen als ihren Kriterien für Open-Source-Lizenzen entsprechend eingestuft (http://www.opensource.org/, Stand 20. Mai 2003). Im Zuge der fortschreitenden Vermarktung der OSS und angesichts des zunehmenden Source-Sharing durch kommerzielle Softwareunternehmen ist mit einer weiteren Verbreitung von Open-Source-Lizenzen zu rechnen, wenn immer mehr Unternehmen und Einzelpersonen festlegen müssen, wie sie ihre Urheberrechte in geeigneter Weise bereitstellen können.

16 Derzeit gibt es in der Softwarebranche keine vergleichbare Bereitstellung von Quellcode für führende Produkte. Zwar liefern viele Anbieter auf Anfrage ihren Kunden Quellcode, doch bieten nur wenige breit angelegte, entsprechende Programme. Dies dürfte sich mit der Zeit ändern, wenn die positiven Auswirkungen von OSS, Shared Source und anderen Programmen zur Bereitstellung von Quellcode allgemeine Anerkennung finden.

Förderung von Lehre und Forschung Quellcode und Dokumentation werden für Schulungen und Lehrmaterial sowie als Grundlage für weiterführende Forschung bereitgestellt.

Förderung neuer Geschäftsmöglichkeiten Lizenzstrukturen und Quellcode werden an Partner bereitgestellt, um für beide Parteien vorteilhafte neue Geschäftsmöglichkeiten zu fördern.

Zum Zeitpunkt des Verfassens dieser Abhandlung wurde für fünf Produktgruppen von Microsoft Quellcode bereitgestellt, der bestimmte Optionen zum Ändern des Codes und Erstellen abgeleiteter Arbeiten umfasst.[17] Drei der Produktgruppen bieten allgemeine Rechte, um abgeleitete Entwicklungen zu erstellen und kommerziell zu vertreiben. Das bedeutet, dass der Programmierer den Code erhält, ändert und unter einer herkömmlichen kommerziellen Binärcode-Lizenz gewinnbringend weitervertreibt, ohne dafür Zahlungen an Microsoft entrichten zu müssen. Alle aktuellen Quellcode-Programme werden von Microsoft kostenlos zur Verfügung gestellt.[18]

6. Aufbau eines Shared-Source-Programms

Für die Einführung der verschiedenen Shared-Source-Programme hat Microsoft erhebliche Ressourcen aufgewandt. Die Bekanntgabe jedes einzelnen Quellcodes basiert auf Entscheidungen, die ein ausgewogenes Verhältnis zwischen Kunden- und Geschäftsvorteilen zum Ziel haben. Zusätzlich haben wir in technische Ressourcen investiert, um erweiterte Tools und Dokumentation zu liefern, die den Nutzen des Quellcode-Zugriffs steigern.

In den Tabellen 1 bis 4 findet sich eine kleine Auswahl der Fragen und Aspekte, die bei Microsoft vor dem Release eines Codes berücksichtigt werden. Diese Übersicht stellt natürlich kein vollständiges Analysetool dar, sondern lediglich ein Beispiel für den Prozess der Entscheidungsfindung, wie er bei einem kommerziellen Softwareanbieter für ein Quellcode-Programm ablaufen kann.

Zusammenfassend lässt sich sagen, dass es für eine durchdacht konzipierte Bereitstellung von Quellcode eine Reihe primärer Zielsetzungen zu erfüllen gilt:

– Es muss ein anschaulicher Einblick in das offengelegte Produkt oder Projekt gegeben werden. Ein Wert entsteht häufig schon allein durch die Zugänglichkeit und Analyse des Codes, ohne diesen zu verändern.

17 Die Windows Academic Shared-Source- und OEM-Lizenzen ermöglichen Forschern das Erstellen zeitweiliger Änderungen des Windows-Quellcodes für Forschungs- und Erprobungszwecke. Alle anderen Gruppen mit Windows-Source-Zugriff (Unternehmenskunden, Systemintegratoren und Regierungsbehörden) besitzen lediglich Rechte zur Einsichtnahme, d. h. sie können den Quellcode zwar anzeigen und debuggen, aber keine abgeleiteten Entwicklungen aus ihm erstellen.

18 Aufgrund der zeitlichen Verzögerung zwischen dem Verfassen und der Veröffentlichung dieses Dokuments wurden keine spezifischen Details zu den Programmen aufgeführt. Weitere Informationen zur Verfügbarkeit von Quellcode von Microsoft unter http://www.microsoft.com/sharedsource/.

Fragen	Aspekte
Welche Community soll diesen Quellcode erhalten?	Nicht alle Quellcode-Releases müssen vollständig und allgemein verfügbar sein. In bestimmten Fällen kann eine Zusammenarbeit mit eingeschränkten Communitys, großen Kunden, wichtigen Partnern oder bestimmten Regierungsbehörden besser geeignet sein.
Worin besteht der Vorteil dieses Quellcodes für diese Unternehmen und Einzelpersonen?	Ein Quellcode deckt nicht alle IT-Belange ab. Daher ist die korrekte Einschätzung seines Wirkungsbereichs ein kritischer Faktor bei der Festlegung der Vorgehensweise für Rechtevergabe und Bereitstellung.
Wie viele Personen werden über den Quellcode verfügen und wie gestaltet sich die Interaktion mit diesen?	Weitreichende Programme können umfangreiche Ressourcen für die Bereitstellung des Quellcodes und/oder die Einbeziehung der Community erfordern. Dabei geht es nicht nur um logistische Belange wie beispielsweise die Download-Kapazität. Es muss u. a. das Ausmaß an technischem Aufwand, Feedback-Verarbeitung und laufenden Investitionen berücksichtigt werden.
Welche geografischen Gebiete kommen für den Quellcode-Zugriff infrage?	Abgesehen von offensichtlichen Aspekten wie der Lokalisierung der Dokumentation, bestehen von Land zu Land auch erhebliche rechtliche Unterschiede, die es zu berücksichtigen gilt. Die Handhabung von Fragen des geistigen Eigentums variiert sehr stark, weshalb sich hier eine spezielle Rechtsberatung empfiehlt.[19]

Tabelle 1: Definition der Zielsetzungen

- Es müssen entsprechende Tools, Dokumentation und Support-Leistungen bereitgestellt werden, um den Wert für die mit dem Code arbeitenden Einzelpersonen zu erhöhen, insbesondere bei Programmen mit Rechten für abgeleitete Entwicklungen.
- Es müssen Methoden für die Verarbeitung von Feedback festgelegt werden, um so Verbesserungen der Code-Basis zu erleichtern.
- Es muss diejenige Community bestimmt werden, die von dem Zugriff auf den Quellcode den größten Vorteil hat.
- Es müssen Rechte festgelegt werden, die den Urheber des Quellcodes sowie alle an ihm arbeitenden Beteiligten schützen.

19 Dies gilt für alle Open-Source-Modelle – OSS, Shared Source u. a. Beispielsweise basieren viele der populärsten OSS-Lizenzen auf im US-amerikanischen Urheberrechtssystem verankerten Konzepten. Da der Code weltweit genutzt wird, variieren die angewandten rechtlichen Grundlagen erheblich.

Fragen	Aspekte
Welcher Quellcode soll verfügbar gemacht werden?	Genau wie es sich bei der Community, für die der Quellcode bereitgestellt werden soll, nicht unbedingt um 100 Prozent der Bevölkerung handelt, muss der bereitgestellte Quellcode auch nicht 100 Prozent eines Produkts darstellen. Bestimmte Bestandteile eines Produkts sind extrem wertvoll, unter Ausschluss einer Offenlegung des Quellcodes an Dritte lizenziert oder von Exportbeschränkungen der jeweiligen Regierung betroffen.
Sind die Rechte zur Offenlegung der jeweiligen Source vorhanden?	Kommerzielle Software enthält oft Bestandteile, die ursprünglich in einem anderen Zusammenhang entwickelt wurden und innerhalb eines umfangreicheren Produkts wiederverwendet oder lizenziert werden. Je mehr Inhaber von Urheberrechten für ein bestimmtes Software–Segment vorhanden sind, desto komplexer wird die Bereitstellung des Quellcodes kommerzieller Produkte.
Wurde der Code für die allgemeine Verwendung bereinigt und wurde die Qualität der Kommentare überprüft?	Innerhalb des Quellcodes finden sich Kommentare der Entwickler, die einen Einblick in deren Gedankengänge liefern sollen. Es kann äußerst hilfreich sein, wenn besonders komplexe Code-Segmente mit Kommentaren versehen sind. Leider enthält der Quellcode häufig sehr lockere Formulierungen, deren öffentliche Bereitstellung erst überprüft werden sollte.
Ist eine Strategie für die Verwaltung des Quellcodes vorhanden, die Bug-Fixes und zukünftige Versionen im Zusammenhang mit der offengelegten Source vorsieht?	Für die meisten erfolgreichen Softwareentwicklungsprojekte bestehen ausgereifte Verwaltungsprozesse in Hinblick auf den Quellcode. Soll ein bislang geschützter Code an eine breitere Basis von Entwicklern bereitgestellt werden, ist ein Prozess erforderlich, um laufende intern durchgeführte Entwicklungsarbeiten für die Community verfügbar zu machen, die nun den Quellcode besitzt.
Wie werden eingereichte Vorschläge oder Code–Fixes gehandhabt?	Genau wie bei der Bereitstellung von neuem intern entwickelten Code muss ein Prozess erstellt werden, der die Entgegennahme von Vorschlägen, Bug-Fixes und neuen Funktionen von Seiten der Community regelt, die Zugriff auf den Code hat. Dabei sollten auch die rechtlichen Konsequenzen und Risiken im Zusammenhang mit der Einbeziehung eingereichter Vorschläge oder Code-Fixes in die Code-Basis erwogen werden.

Tabelle 2: Verwaltung des Quellcodes

Fragen	Aspekte
Welche Rechte sollen für Anzeige, Debugging, Änderung und Vertrieb vergeben werden?	Obgleich der rechtliche Aspekt bei der Festlegung von Lizenzierungsrechten für Quellcode eine wichtige Rolle spielt, kommt es doch vor allem auf die Wünsche der Kunden und Partner an. Entscheidend ist, wie diese von dem Quellcode profitieren können. Auch die treuhänderische Verantwortung gegenüber den Aktionären im Hinblick auf den Schutz des geistigen Eigentums ist ein kritischer Faktor, der allerdings erst sekundär zum Tragen kommen sollte. Ein erfolgreiches Quellcode-Lizenzierungsprogramm schafft einen Ausgleich zwischen den genannten Faktoren zum Wohle aller Beteiligten.
Kann ein kommerzieller Weitervertrieb auf Basis eingeräumter abgeleiteter Rechte erfolgen?	Microsoft hat sich dafür entschieden, zwei verschiedene Ansätze hinsichtlich der Lizenzen für abgeleitete Entwicklungen zu implementieren, die unterschiedliche Geschäftsziele innerhalb unseres Programms zur Lizenzierung von Quellcode widerspiegeln. Die kommerzielle Nutzung abgeleiteter Entwicklungen ist ein zentraler Punkt in der Bereitstellung von Quellcode durch Anbieter kommerzieller Software.

Tabelle 3: Lizenzierung

7. Erfahrungen und Prognosen

Die wichtigste Erkenntnis, die wir aus dem Shared-Source-Prozess gewonnen haben, ist die Tatsache, dass der Quellcode eine Produkteigenschaft darstellt. Diese Eigenschaft wird zwar von vielen Anwendern nie genutzt werden, doch bedeutet allein schon die Möglichkeit ihrer Nutzung einen Mehrwert. Unsere Kunden und Partner, die über den Quellcode verfügen und diesen aktiv nutzen, berichten, dass er einen unschätzbaren Mehrwert bei der Verwendung unserer Produkte darstellt. Bisher stellt diese Gruppe allerdings im Vergleich zu allen Einzel- und Unternehmensanwendern von Microsoft-Produkten einen verschwindend kleinen Anteil dar.

Die Shared-Source-Initiative besteht nun seit drei Jahren, wobei Microsoft jedoch bereits seit mehr als 13 Jahren Quellcode an akademische Institutionen und OEMs liefert. Vor 2001 lief der Austausch von Quellcode nur mit einem begrenzten Teilnehmerkreis und weniger formell als heute ab. Wir haben jedoch aus dem Dialog mit unseren Kunden sowie aus OSS-Modellen gelernt, dass es bei dieser Initiative vor allem um das richtige Verhältnis geht. In vieler Hinsicht befindet sich Shared Source immer noch seiner Version-2.0-Phase. Der Erfolg der bisherigen Programme hat uns gezeigt, wie wichtig eine Ausweitung der Initiative auch für anderen Code ist.

In absehbarer Zukunft wird die Lizenzierung von Quellcode auch weiterhin ein heikles Thema in der Branche darstellen. Grundlegende Fragen zur Rolle des geistigen

Eigentums für zukünftige Innovationen sind zu stellen. Im Zentrum der Shared–Source-Initiative steht die Annahme, dass das geistige Eigentum und dessen Schutz die Grundlage für den Erfolg eines fortwährenden Innovationskreislaufs sind. Die Bekanntgabe von Quellcode hat zwar sehr viele Vorteile, sie darf aber nicht zum Ausschluss einer erfolgreichen Softwarebranche führen. Darüber hinaus kann sie nicht als Wundermittel für alle Belange der Informationstechnik gesehen werden. Es werden daher auch weiterhin verschiedene Modelle nebeneinander existieren. Shared Source ist Teil eines großen Mosaiks, wobei wir uns glücklich schätzen, ein paar von dessen Steinchen zusammenfügen zu dürfen.

Fragen	Aspekte
Wie erfolgt die Bereitstellung des Quellcodes? Wird einfach ein Quellcode-Paket veröffentlicht oder entsteht ein Mehrwert durch ein Bereitstellungstool?	Es gibt zahlreiche Möglichkeiten für die Bereitstellung von Quellcode. Die OSS-Community verfügt über eine Reihe von Websites wie z. B. SourceForge von VA Software Corp., um Quellcode bereitzustellen und Projekte verwalten zu können. Ein Unternehmen kann seinen Quellcode aber auch selber bereitstellen, wie Microsoft dies mit seinem Projekt GotDotNet WorkSpaces getan hat. Microsoft hat zudem eine sichere Web-Infrastruktur für die Bereitstellung von Windows-Quellcode erstellt: MSDN Code Center Premium. Andere Gruppen innerhalb von Microsoft haben sich für einfache Web-Downloads von Source-Dateien entschieden, wobei die Wahl des Toolsets und der Entwicklungsumgebung dem einzelnen Entwickler überlassen bleibt. Die Bestimmung des Umfangs der beteiligten Community sowie des Volumens des in dem jeweiligen Release enthaltenen Quellcodes ist ein wichtiger Faktor für die Wahl der Bereitstellungsmethode.
Wie soll die Community aus Einzelpersonen und Unternehmen mit Zugriff auf den Quellcode einbezogen werden?	Auch wenn eine Community aus Entwicklern sich für eine bestimmte Source-Basis spontan zusammenfinden kann, umfassen erfolgreiche Programme dennoch eine Beteiligung der für die Erstellung des Codes verantwortlichen Entwickler. Darüber hinaus sorgt ein durchdachtes Projektmanagement dafür, dass die Community auf produktive Weise einbezogen wird.
Wurde eine zusätzliche Support-Dokumentation für die Source–Basis erstellt?	Das Erstellen einer hochwertigen Softwaredokumentation hat sich als eines der teuersten und schwierigsten Probleme in dieser Branche erwiesen. Je mehr Informationen den mit einer bestimmten Source-Basis arbeitenden Entwicklern bereitgestellt werden, desto besser. Obwohl dies nicht zwingend erforderlich ist, wird dadurch sicherlich die Qualität des Source-Sharing-Programms als Ganzes verbessert.

Tabelle 4: Umsetzung

Coases Pinguin beginnt zu fliegen – Der institutionelle Wandel in der Softwareindustrie

MATTHIAS BÄRWOLFF

Dieser Artikel behandelt den Wandel der Prozesse und Strukturen in der kommerziellen Softwareindustrie bedingt durch den wachsenden Einfluss von Open Source als „Produktionsparadigma". Der zu beobachtende institutionelle Wandel lässt sich als effizienzsteigernde Anpassung der Produktions- und Nutzungsstrukturen im Bereich Software begreifen.

1. Einleitung

Als Coase 1937 in seinem bahnbrechenden Artikel „The nature of the firm" die Existenz der Firma als ökonomische Reaktion auf die „Kosten der Nutzung des Preissystems des Marktes" – seither Transaktionskosten genannt – zurückführte, erschuf er die bis heute allgemein akzeptierten ökonomischen Grundmodi Markt und Firma. Dieser Artikel will der häufig aufgeworfenen Frage nach der Einordnung von Open Source relativ zu diesen beiden Modi nachgehen.

Einerseits kann man die Entwicklung von Open-Source-Software (OSS) im analytischen Rahmen der *Firma* auffassen, bei dem die Beziehungen der Mitwirkenden durch Open-Source-Lizenzverträge geprägt sind und eine Struktur durch „psychologische Verträge" (Rousseau 1989) geschaffen wird. Anderseits wird jedoch oft auch der *Markt* als Analogie für die „offene Evolution" bei Open-Source-Software verwendet.[1] Genau genommen passt Open Source eigentlich in keines der beiden Raster, tatsächlich bewegt es sich inner- und außerhalb des Kontinuums Markt-Firma, ohne sich dabei allzu leicht festlegen zu lassen.

Der Titel wie auch das Thema dieses Artikels sind weitgehend motiviert durch Benklers Artikel „Coase's Penguin, or, Linux and *The Nature of the Firm*" (2002). Dieser nutzt Coases Transaktionskostentheorie[2] als methodologischen Hintergrund für seine Analyse und befasst sich mit der Frage, warum Open Source als Produktionsparadigma so erfolgreich sein kann, wenn weder Preissignale noch Anweisungen für eine

1 Siehe der Beitrag von Bauer und Pizka in diesem Buch. Ökonomisch gesehen macht die von ihnen aufgegriffene Analogie nur bedingt Sinn, da bei der Entwicklung von OSS keine Preissignale zur Steuerung der Ressourcenverteilung zum Tragen kommen.

2 In der englischsprachigen Literatur findet man hierfür den Begriff *transaction costs theory* (TCE).

effiziente Ressourcenverteilung zum Tragen kommen. Benkler schließt daraus, dass es sich bei Open Source um einen „entstehenden neuen dritten Modus der Produktion" handelt. Demil und Lecocq (2003) nennen diesen Modus, der sich durch die Bedingungen von Open-Source-Lizenzen konstituiert, *„bazaar governance"*, in Anlehnung an Raymonds berühmten Aufsatz „The Cathedral and the Bazaar".

Es ist unzweifelhaft, dass dieser „dritte Modus" 2005, also drei Jahre später, die kommerzielle Softwareindustrie erfasst hat.[3] Selbst Firmen wie Microsoft, die traditionell fast keine strategischen Überschneidungen mit Open Source haben, erkennen dessen Bedeutung und befassen sich intensiv damit, ihr Geschäftsmodell zumindest teilweise den veränderten Rahmenbedingungen anzupassen.[4] Der Wandel in der Softwareindustrie, bedingt durch Open-Source-Software, ist also unübersehbar. Dieser Artikel soll sich also der Frage widmen, wie man diesen Wandel ökonomisch begreifen und sinnvoll fassen kann. Hierzu wird ein Blick auf die relevanten ökonomischen Institutionen sowie deren Evolution und Effizienz geworfen. Speziell wird der Frage nachgegangen, wie Open Source die ökonomische Rationalität der kommerziellen Akteure beeinflusst und verändert.

2. Institutionen im Bereich Software

Die wichtigsten Institutionen[5] im Bereich von Softwareentwicklung und -nutzung, und damit auch der Transaktionen, die gemeinhin mit Software in Verbindung stehen, sind rechtlicher und technischer Natur. Nicht ohne Grund konzentriert sich die aktuelle Diskussion im Bereich „Internet Governance" primär auf rechtliche und technische Regulierungsaspekte bei Software. Die Bedeutung rechtlicher Rahmenbedingungen wird schon alleine durch die konstituierende und äußerst umstrittene Rolle von Urheberrecht und Patentrecht für Software deutlich. Lessig (1999) hat außerdem die generell zunehmende Bedeutung von *Code* als „regulierende Instanz", also als Institution, identifiziert.

Beide institutionellen Bereiche, Recht und Code, werden, sowohl für sich selbst als auch in ihren Wechselwirkungen, in der aktuellen Literatur relativ umfassend behandelt. Dabei wird primär auf die tatsächlichen und potentiellen negativen Effekte für die Verbraucher und die allgemeine Wohlfahrt eingegangen.[6] Zumeist wird bei diesen Betrachtungen das bestehende Regime der Institutionen des „geistigen Eigentums"

3 So bemerkt ein Analyst treffend: „The impact of open source in the enterprise is undeniable" (Scot Petersen, „Open Source Is Showing Its Value", eWeek, 14. Februar 2005, http://www.eweek.com/article2/0,1759,1763034,00.asp).

4 Siehe der Beitrag von Seemayer und Matusow in diesem Buch.

5 Zur Definition von Institutionen sagt North (1990, S. 4): „Institutions define and limit the set of choices of individuals. Institutional include any form of constraint that human beings devise to shape human interaction. They can be [. . .] both [. . .] formal [. . .] and [. . .] informal." Siehe zum Begriff Institution auch einen der maßgeblichen Begründer der Institutionenökonomie Commons (1931).

6 Lessig (1999) hat in diesem Zusammenhang insbesondere das Argument populär gemacht, dass Regulierung durch Code ohne Einflussmöglichkeiten für die Verbraucher, wenn man so will, „hinter ihrem Rücken" stattfindet und dadurch letztendlich einen negativen Einfluss auf die allgemeine Wohlfahrt hat. Als Beispiel führt er unter anderem die derzeit zu beobachtenden Abweichungen vom ursprünglichen *End-to-End*-Design des Internets an.

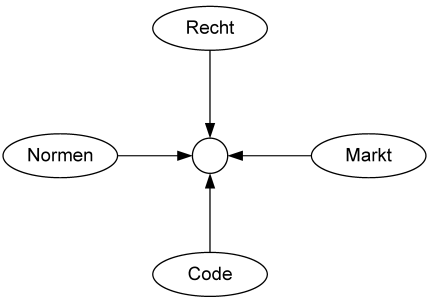


Abbildung 1: „Punkt-Modell" von Lessig (1999)

als inadäquat und hinderlich für den technischen Fortschritt, die günstige Entwicklung im Bereich *Code* also, identifiziert, der speziell in diesem Bereich auf sequentiellen und verteilten Innovationen beruht.[7] Der institutionelle Einfluss von Code als Einschränkung von Handlungsspielräumen von Akteuren wird also erkannt, die Lösung des daraus folgenden Problemkomplexes jedoch praktisch ausschließlich im Bereich des Rechts gesehen.

Angesichts der aktuell zu beobachtenden Entwicklungen stellt sich jedoch die Frage, ob die Institution Recht tatsächlich den entscheidenden strukturellen Einfluss auf Softwareentwicklung und -nutzung hat, wie dies bislang unterstellt wurde, oder ob sich dieser im Zuge eines institutionellen Wandels verringert. Um uns einer möglichen Antwort zu nähern, ist es sinnvoll, in Anlehnung an Commons (1931) zwei weitere institutionelle Bereiche zu betrachten: *Ökonomie*, oder besser *wirtschaftliche Rationalität*, und *Ethik*, oder auch Normen. Lessig (1999) hat in „Code and Other Laws of Cyberspace" diesen Gedanken mit einem einfachen „Punkt-Modell" aufgegriffen (Abbildung 1). In diesem finden sich Code, Recht, Markt und Normen als „regulierende Modalitäten".[8] Lessig betont, dass die einzelnen „Regularien", oder, wenn man so will, Institutionen, sich gegenseitig beeinflussen und in diesem Prozess einer Evolution unterliegen. Dieser Gedanke greift unter anderem auf North (1990) zurück, der deutlich gemacht hat, dass Organisationen und Institutionen sich wechselseitig beeinflussen.

In Abbildung 2 versuchen wir, das Zusammenspiel der Institutionen im Bereich Softwareentwicklung und -nutzung so darzustellen, wie wir es insbesondere im Hinblick auf die zentrale Rolle von *Code* im Folgenden betrachten wollen. Letztlich konzentrieren wir uns in diesem Modell auf die Wechselwirkungen von Code mit anderen Institutionen, speziell mit *ökomischer Rationalität*. Dabei begreifen wir Code sowohl als Institution als auch als ökonomisches Gut, weil es als solches einen signifikanten Ein-

7 Siehe hierzu die Beiträge von Bessen und Maskin sowie von Hippel in diesem Buch.

8 Lessigs Modell ist in sofern sehr allgemein, als dass der konkrete Gegenstand der Regulierung weitgehend im Unklaren bleibt. Auch die Komponente „Markt" ist sehr unspezifisch. Die Beispiele, die Lessig zu deren Illustration heranzieht, deuten vielmehr auf den Aspekt der Preisregulierung durch Steuern als „regulierende Modalität" hin.

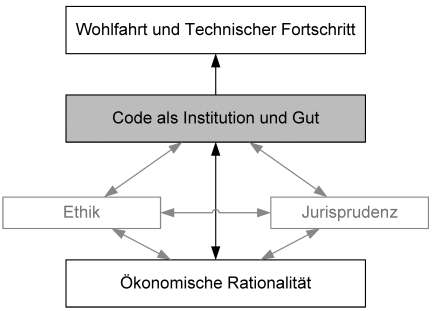


Abbildung 2: Die Beziehung von Code und ökonomischer Rationalität

fluss auf ökonomische Wohlfahrt und Fortschritt hat. Der Charakter von Code ist also ein dualer, zum einen hat er Einfluss auf Organisationen und andere Institutionen, zum anderen beeinflusst er die allgemeine Wohlfahrt und den technischen Fortschritt.

3. Die Effizienz der Institutionen im Hinblick auf *Code*

Praktisch seit der Einführung von Urheber- und Patentrecht wurden deren institutionelle Auswirkungen auf die Entwicklung und Nutzung von Software kritisiert.[9] Dabei bezieht sich die Kritik an den bestehenden Institutionen nicht allein auf statische Wohlfahrtserwägungen, sondern primär auf das Argument, dass technologischer Fortschritt basierend auf sequentiellen Innovationen durch die bestehenden Institutionen erschwert und behindert wird.[10] Insbesondere wurde von vielen Beobachtern angemerkt, dass die ökonomische Rationalität, bedingt durch Urheber- und Patentrecht, einen negativen Einfluss auf die sequentielle Entwicklung im Bereich Software hat: Die Maximierung privater Interessen mittels Durchsetzung von Ausschließlichkeitsrechten, gewährt durch das Urheber- und Patentrecht, steht in unauflösbarem Konflikt mit den Interessen der Allgemeinheit.

Während die empirischen Befunde für die Gültigkeit dieser Argumentationen lange Zeit ausstanden (Liebowitz und Margolis 1999), stellt spätestens der Erfolg von Open-Source-Software die bestehenden rechtlichen Institutionen hinsichtlich ihrer Effizienz ernsthaft in Frage. Der Bestand einer solchen Ineffizienz wird jedoch nach North (1990) zumindest teilweise bedingt durch das Wechselspiel zwischen bestehenden Institutionen und Organisationen, die einem institutionellen Wandel entgegenwirken. Es ist also mithin schwierig, bestehende Ineffizienzen aufzulösen, wenn Organisationen wie große Firmen davon profitieren.

Die meisten Betrachtungen beklagen zwar eben diesen Umstand, übersehen jedoch den zunehmenden institutionellen Charakter von Code, speziell in der Ausprägung

9 Siehe z. B. Samuelson (1984).
10 Siehe der Beitrag von Bessen und Maskin in diesem Buch.

Open Source, für die ökonomische Rationalität der Beteiligten.[11] Man kann Open Source, im Sinne von Romer (1993), als „Meta-Idee" begreifen, die die ökonomische Rationalität bei der Produktion, Nutzung und Weitergabe von Code grundlegend verändert und damit qualitativ neues, zuvor nicht da gewesenes Wachstum erzeugt.[12] Diese veränderte Rationalität wäre Ausdruck der „adaptiven Effizienz" (North 1990), die Reaktion auf veränderte Gegebenheiten durch die Existenz von Open Source.

Tatsächlich verdichtet sich dieser Eindruck, wenn man die aktuellen Markttendenzen beobachtet. So bemerkt beispielsweise John Terpstra, einer der Hauptentwickler von Samba und Redner auf der diesjährigen *LinuxWorld* in Boston: „Whatever the outcome of current IP [intellectual property] litigation involving Linux and open source, the market will move forward".[13] Damit deutet er auf eben diesen Trend hin: Open Source spielt für immer mehr kommerzielle Firmen eine bedeutende Rolle, die unweigerlich den Kern ihres Geschäftsmodells berührt.

4. Der Einfluss von Open Source auf *ökonomische Rationalität*

Angesichts der steigenden Relevanz von Open-Source-Software für kommerzielle Unternehmen, nicht nur bezüglich deren Anwendung, sondern insbesondere auch der Mitwirkung an deren Produktion, sollte es möglich sein, ökonomische Erklärungsmuster für diesen Wandel zu identifizieren.

Open Source wird als „Produktionsparadigma" vollständig definiert durch Open–Source-Lizenzen.[14] Mit solchen Lizenzen wird eine Institution erzeugt, die hauptsächlich vertragsrechtliche, zu einem gewissen Grade auch ethische, Rahmenbedingungen für die Schaffung, Nutzung und Weitergabe von Open-Source-Software festlegt.

Es wird häufig unterstellt, dass Open Source für kommerzielle Firmen, oder überwiegend „extrinsisch motivierte Akteure" nur von marginalem Interesse sein kann, da Open-Source-Software selbst eine direkte Aneignung von Mehrwert durch Transaktionen basierend auf herkömmlichen exklusiven Eigentumsrechten scheinbar ausschließt. Schließlich ist Exklusiveigentum an Open-Source-Software durch die Lizenzbedingungen, der sie unterliegt, praktisch unmöglich. So stellen beispielsweise Osterloh et al. (2004) in Anlehnung an Fehr und Falk (2001) die These auf, dass das Vorhandensein von „intrinsischer Motivation" eine unabdingbare Voraussetzung für erfolgreiche Open-Source-Projekte darstellt.[15] Und auch Weber (2004) widmet sich

11 Die institutionelle Bedeutung von Open Source wird sehr gut illustriert durch die teilweise haarsträubend scharfen Attacken von Microsoft gegen Open Source und speziell die GPL am Anfang des 21. Jahrhunderts.

12 So bemerkt Paul Romer: „Open Source erinnert uns daran, dass der Markt nicht die einzige Lösung ist" (zitiert in Ludwig Siegele, „Lieber Ruhm im Netz als Rubel im Sack", Die Zeit, Dezember 2000, http://zeus.zeit.de/text/archiv/2000/12/200012.open_content_.xml).

13 Jan Stafford , „LinuxWorld preview: Open source rules, SCO fades and apps abound", SearchEnterpriseLinux.com, 9. Februar 2005, http://searchenterpriselinux.techtarget.com/originalContent/0,289142,sid39_gci1052513,00.html

14 Siehe die Webseite der *Open Source Initiative* (OSI, http://www.opensource.org) zu den verschiedenen existierenden Open-Source-Lizenzen.

15 Es soll hier angemerkt werden, dass die Dichotomie extrinsische und intrinsische Motivation keineswegs

in seiner exzellenten Abhandlung über die Wirkungsweise von „klassischen“ Open–Source-Projekten, wie Linux und Apache, nur am Rande der Bedeutung kommerzieller Akteure für Open Source.

Es mehrt sich jedoch die empirische Evidenz dafür, dass Reputationseffekte und damit einhergehende Erhöhungen des individuell erzielbaren Lohnniveaus für einzelne Entwickler sehr wohl eine wichtige extrinsische Motivation für die Mitarbeit an Open-Source-Software sein können (Lerner und Tirole 2000, Hann et al. 2004). Außerdem lässt sich die unübersehbare Mitwirkung von kommerziellen Firmen zweifellos *nur* „extrinsisch“ erklären. Es scheint mithin praktikabel zu sein, wenigstens ökonomische Rationalität im Sinne von Jevons (1923) für unsere Ausführungen anzunehmen.[16]

Warum also wirken kommerziell motivierte Firmen, denen wir ökonomisch rationales Verhalten unterstellen, an Open-Source-Entwicklung mit? Eine tragfähige und weitreichende Erklärung, die auch Boyle (2000) aufgegriffen hat, ist, dass Informationsgüter nur selten in vollständiger Isolation von dinghaften Gütern oder nachgelagerten Dienstleistungen auftreten:[17]

> „Information goods do not exist in isolation. The good [. . .] comes “bundled” with a large number of other customer needs, social implications, market effects, and business opportunities. These linked or bundled phenomena may well be excludable to a greater degree than the information good itself.“ (S. 2015)

Koenig (2004) elaboriert entsprechende aktuelle Open-Source-Strategien von Softwarefirmen und führt als Beispiele unter anderem Firmen wie Zope, JBoss und Red Hat an, die ihre Softwareprodukte als Open-Source-Software handhaben, Dienstleistungen wie Anpassung, Beratung, Schulung oder Zertifizierung jedoch als kommerzielle Dienstleitungen anbieten.

Open Source kann also, wenn wir den Bogen von der Diskussion von Geschäftsmodellen etwas weiter spannen, als Infrastruktur aufgefasst werden, auf der weitere Produkte oder Dienstleistungen aufsetzen.[18] Die Erkenntnis, dass viele Softwarepro-

---

so trivial ist, wie es auf den ersten Blick scheinen mag. Viele scheinbar intrinsische Handlungen von Personen schließen extrinsische Motivation nicht aus.

16 Es kann zweifellos als gänzlich unproblematische Annahme gelten, dass Akteure in ökonomischen Systemen im Allgemeinen bessere Situationen den schlechteren vorziehen. Diese Annahme ist unabhängig von der empirischen Beobachtung, dass es keine vollkommenen Märkte gibt, auf denen sämtliche Akteure zu jedem Zeitpunkt vollständig informiert sind und in unendlich kurzer Zeit ihren Präferenzen entsprechend optimale Transaktionen tätigen.

17 Siehe auch Hirshleifer (1971) für eine frühere Elaboration.

18 Michael Tiemann („How Linux will Revolutionize the Embedded Market“, Linuxdevices.com, 22. Mai 2002, http://linuxdevices.com/articles/AT7248149889.html) trifft in diesem Zusammenhang eine wichtige Unterscheidung zwischen Open-Source-Software als Projekt und Open-Source-Software als Plattform:

> „Projects are organized around specific, though possibly far-reaching, technical goals (such as running on a laptop, in a cluster, in a router, on a mainframe, on a particular chip architecture, etc.). A platform, on the other hand, provides a common environment that can be widely deployed and serviced. Linux itself is a project (it moves forward largely

dukte letztlich Plattformen bzw. Infrastrukturen begründen, ist fundamental für das Verständnis von Wertschöpfung im Bereich Software. Nicht nur sind Anwendungen von Infrastrukturen im Vorhinein weitgehend unspezifiziert und offen, es ist sogar möglich, dass Software als Infrastruktur für weitere Infrastrukturen dient, etwa eine Datenbank oder ein *Content Management System* auf einem Betriebssystem.

Frischmann (2004) führt in seiner Elaboration zu den ökonomischen Eigenschaften von Infrastrukturen aus, dass eben durch die Unspezifiziertheit und damit Unvorhersehbarkeit der Nutzungen ein auf privaten Exklusivrechten und Markttransaktionen basierendes Eigentumsregime ökonomisch ineffizient ist. Effizienter ist es, Frischmann zufolge, solche Infrastrukturen als Allmenden, engl. *Commons*, zu verwalten. Er greift damit einen Gedanken auf, den erstmals Rose (1986) in dieser Form postuliert hat.[19] Zope, JBoss und RedHat zeigen die kommerzielle Sinnhaftigkeit eines solchen Vorgehens und damit dessen ökonomische Rationalität.

Ein in diesem Zusammenhang weiterer wichtiger Faktor, der die Entstehung von Allmendegütern gerade bei Softwareprodukten mit Infrastruktureigenschaften begünstigt, ist deren zunehmender *Commodity*-Charakter. Eine Software, deren Schnittstellen und Funktionalitäten weitgehend spezifiziert und standardisiert sind, ist relativ leicht nachahmbar.[20] Da die Grenzkosten für Software bekanntermaßen bei Null liegen, gehen bei hinreichender Wettbewerbsintensität die Margen und der Gleichgewichtspreis für solche Güter ebenfalls gegen Null. Mithin schwinden dadurch die Opportunitätskosten[21] für eine Firma, ihr Softwareprodukt als Open-Source-Software zu handhaben.[22]

Die zwei genannten Faktoren – der höhere Nutzen auf der Nachfrageseite durch die Handhabung von Infrastrukturen als Allmendegüter und die vernachlässigbaren Grenzkosten, die den Gleichgewichtspreis und damit die Opportunitätskosten von Open Source gegen Null treiben – bedingen zu einem gewissen Grade einen dritten

for technical reasons), whereas commercial Linux distributions are platforms (which move forward more for commercial reasons).“

19 Die akademische Diskussion um die Effizienz und Nachhaltigkeit von Allmende-basierten Eigentumsregimen ist, nicht zuletzt inspiriert durch das Internet und Open Source, sehr lebhaft. Benkler (2004), der schon 2002 einen als klassisch geltenden Beitrag zur Diskussion geleistet hat, spricht in seiner empirisch geprägten Diskussion der Übertragbarkeit von *peer production* auf dinghafte Güter von einem entstehenden „dritten Modus der organisierten ökonomischen Produktion“ (S. 357). Der Beitrag von Lutterbeck in diesem Buch befasst sich mit der Schwierigkeit, die relevanten Gesichtspunkte der Thematik schlüssig zusammenzuführen. Seiner Meinung nach darf sich die Diskussion nicht nur mit Ökonomie befassen, sondern muss sich zudem die Ingenieurswissenschaften berücksichtigen.

20 Diese Argumentation ist natürlich sehr vereinfacht. Es gibt durchaus komplexe Schnittstellen deren Nachahmung nicht trivial ist, insbesondere wenn deren Spezifikation nicht verfügbar ist, etwa bei einem proprietären Betriebssystem wie *Microsoft Windows*. Jedoch bejaht die Rechtssprechung prinzipiell die Nachahmung von Schnittstellen zum Zwecke der Interoperabilität und Konkurrenz. So kann ein Konkurrent ein System mit ähnlicher Benutzerschnittstelle und Funktionalität auch dann anbieten, wenn die internen Schnittstellen des nachgeahmten Systems unbekannt sind. Siehe hierzu auch Samuelson und Scotchmer (2002).

21 Opportunitätskosten bezeichnet die Kosten, die durch den Verzicht auf eine nahe liegende Alternative entstehen – in diesem Falle: die Weiterführung der Software als proprietäre Software.

22 Als Beispiel sei hier die Freigabe des Quelltextes von Netscapes *Netscape Navigator* 1998 unter einer Open-Source-Lizenz genannt. Siehe hierzu auch der Artikel von Bauer und Pizka in diesem Buch.

Faktor (Atul Chitnis 2004, pers. comm.): Die Nachfrage nach Open Source steigt, weil sowohl der Gebrauchswert einer Software als auch die Konsumentenrente für den Anwender höher ist, wenn es sich um eine Open-Source-Software handelt. So erhöht sich bei Spezialsoftware das Spektrum an Optionen und damit der potenzielle Wert der Software durch die Modifizierbarkeit des Quelltextes. Bei Standardsoftware steigt ob der geringen Transaktionskosten durch Open Source die Verfügbarkeit einer Software, was wiederum eine höhere Wertschöpfung auf der Nachfrageseite bedingt.

Open Source gewinnt nicht nur an Relevanz durch nachfrageseitige Effekte, sondern auch durch produktionsseitige Effekte. Benkler (2002) widmet sich der Effizienz der Produktion von Open Source und zeigt in Elaborierung von Coases (1937) Transaktionskostenansatz, dass es ökonomisch rational sein kann, ein Produkt zusammen mit anderen unter Aufgabe von exklusiven Eigentumsrechten zu erzeugen. Er nennt diesen Ansatz *commons-based peer production* und argumentiert instruktiv, dass diese Art von Produktion insbesondere Informationskosten effektiver senkt, als dies bei den ökonomischen Produktionsmodi Markt oder Firma der Fall ist. Obgleich hier weder Preissignale, wie im Markt, noch transaktionskostensenkende Anweisungen, wie in einer Firma, zum Tragen kommen und man damit Ineffizienz unterstellen könnte (Kooths et al. 2003), machen die Kostenvorteile durch die Senkung von Informationskosten *commons-based peer production* in einigen Belangen zu einem effizienteren Modus der Produktion als Markt oder Firma.[23]

Dem lässt sich hinzufügen, dass es sich schlüssig und mit einiger empirischer Fundierung argumentieren lässt, dass es in vielen Open-Source-Projekten durchaus *managerial direction* gibt, aber eben nicht durch Vorstände oder Geschäftsführer, sondern durch Komitees oder dergleichen.[24] Außerdem steht die empirische Evidenz für die Effizienz von preisgesteuerten Marktmechanismen im Bereich Software angesichts der unbestreitbar enormen Transaktionskosten weitgehend aus.[25]

Während also Frischmann argumentiert, dass ein Allmendegut zu einem größeren nachfrageseitigen Nutzen führen kann und damit indirekt Nutzen und somit kommerzielle Motivation für Produzenten nach sich ziehen kann (Beratung, Schulung, Anpassung etc.), identifiziert Benkler Effizienzgewinne bei der Produktion auf der Angebotsseite als Faktoren, die eine ökonomische Sinnhaftigkeit von Allmende-basierter Produktion bedingen können.[26] Open Source macht also nicht nur ökonomischen Sinn für Nachfrager, sondern auch für Anbieter, indem die Wertschöpfung von der Software selbst auf angrenzende Bereiche hin verlagert wird.

23 Siehe zu einer Elaboration der Vor- und Nachteile auch Demil und Lecocq (2003).

24 Siehe wiederum der Artikel von Bauer und Pizka im Kapitel Technik und die dortigen Referenzen.

25 Insbesondere durch die relativ leichte digitale Kopierbarkeit von Software sind die Transaktionskosten aus der Durchsetzung von Eigentumsrechten immens. Es ist zumindest denkbar, dass die Kosten der Durchsetzung solcher Rechte die der Aufgabe bzw. Abgabe solcher Rechte, beispielsweise in Form von Open Source, übersteigen. Siehe zur Diskussion um die Effizienz von Marktmechanismen im Bereich Software auch die Replik von Pasche und von Engelhardt (2004) auf Kooths et al. (2003).

26 Von Hippel und andere Innovationsforscher gehen in ihrer Analyse von Open Source noch einen Schritt weiter und stellen die Dichotomie Produzent-Nutzer gleich gänzlich in Frage. Dabei verweisen sie unter anderem auf anekdotische Evidenz aus der Geschichte der Produktion von Surfbrettern. Die Reichweite dieses Ansatzes lässt sich jedoch noch nicht vollständig absehen.

5. Fazit

Open Source hat unzweifelhaft die Strukturen in der Softwareindustrie verändert. So bemerken Demil und Lecocq: „[I]ndustries are not constrained to adopt only pre-existing forms of organization" (2003, S. 21). In ihrer Elaboration der *governance structure*, die durch Open-Source-Lizenzen geschaffen wird, kommen Demil und Lecocq zu dem Schluss, dass Firmen Open Source als strukturierendes Element ihrer Beziehungen zu anderen Firmen wählen, wenn sie sich strategischen Vorteile davon versprechen, Netzwerkeffekte durch die Schaffung von Standards zu erzeugen und für sich auszunutzen.

Während dies sicher ein Teil der Erklärung für die ökonomische Tragfähigkeit von Open Source ist, glauben wir, dass Open-Source-Software einen so fundamentalen Wandel in der Softwareindustrie bedingt, weil sie den effizienteren Modus der Produktion begründet und einen höheren absoluten Mehrwert erzeugt. Dies ist insbesondere der Fall für Infrastruktur-Güter und in ihrer Funktionalität weitgehend spezifizierte Güter.

Wenn es also einen ökonomischen wie auch technischen Imperativ für Interoperabilität von Komponenten in Softwaresystemen gibt, so hat dieser zur Folge, dass Open Source als strukturierendes Prinzip die Softwareindustrie nachhaltig beeinflusst. Dabei setzen sich nicht nur die Komponenten, die auf essenzielle Infrastrukturen zugreifen, einem wohlfahrtssteigernden Wettbewerb aus, sondern eben auch die Infrastrukturen selbst. Die nächsten Jahre sollten zeigen, ob und wie sich diese Prognose bewahrheitet.

Literaturverzeichnis

Benkler, Y. (2002), 'Coase's penguin, or, Linux and The nature of the firm', *Yale Law Journal* **112**(3), S. 369–446.

Benkler, Y. (2004), 'Sharing nicely: On shareable goods and the emergence of sharing as a modality of economic production', *Yale Law Journal* **114**(2), S. 273–358. http://www.yalelawjournal.org/pdf/114-2/Benkler_FINAL_YLJ114-2.pdf.

Boyle, J. (2000), 'Cruel, mean, or lavish? Economic analysis, price discrimination and digital intellectual property', *Vanderbilt Law Review* **53**(6), S. 2007–2039.

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

Commons, J. R. (1931), 'Institutional economics', *American Economic Review* **21**, S. 648–657.

Demil, B. und Lecocq, X. (2003), 'Neither market nor hierarchy or network: The emerging bazaar governance', http://opensource.mit.edu.

Fehr, E. und Falk, A. (2001), 'Psychological Foundations of Incentives'. Working Paper No. 95. 2001.

Frischmann, B. M. (2004), 'An economic theory of infrastructure and sustainable infrastructure commons', SSRN Electronic Library, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=588424.

Hann, I.-H., Roberts, J., Slaughter, S. und Fielding, R. (2004), 'An Empirical Analysis of Economic Returns to Open Source Participation', Faculty Development Grant und Carnegie Bosch Institute, Carnegie Mellon University.

Hirshleifer, J. (1971), 'The private and social value of information and the reward to inventive activity', *American Economic Review* **61**, S. 561.

Jevons, S. W. (1923), *Die Theorie der politischen Ökonomie*, Heinrich Wäntig, Leipzig.

Koenig, J. (2004), 'Open Source business strategies', (überarbeitete Version) http://www.riseforth.com/images/Seven%20Strategies%20-%20Koenig.pdf.

Kooths, S., Langenfurth, M. und Kalwey, N. (2003), Open-Source-Software. Eine volkswirtschaftliche Bewertung, Economic Research Studies 4, Muenster Institute for Computational Economics, Münster. http://mice.uni-muenster.de.

Lerner, J. und Tirole, J. (2000), 'The Simple Economics of Open Source', *Journal of Industrial Economics* **52**, S. 197–234. NBER Working Paper, Nr. 7600.

Lessig, L. (1999), *Code and other laws of cyberspace*, Basic Books, New York.

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

North, D. C. (1990), *Institutions, institutional change and economic performance*, Cambridge University Press, Cambridge.

Osterloh, M., Rota, S. und Kuster, B. (2004), Open-Source-Softwareproduktion: Ein neues Innovationsmodell, *in* B. Lutterbeck und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 121–138.

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

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

Romer, P. M. (1993), Economic Growth, *in* D. R. Henderson (Hrsg.), 'The Fortune Encyclopedia of Economics', Warner Books, New York.

Rose, C. (1986), 'The comedy of the commons: Custom, commerce, and inherently public property', *University Chicago Law Review* **53**.

Rousseau, D. M. (1989), 'Psychological and implied contracts in organisations', *Employee Responsibilities and Rights Journal* **2**(2), S. 121–139.

Samuelson, P. (1984), 'CONTU revisited: The case against copyright protection for computer programs in machine-readable form', *Duke Law Journal* S. 663–769.

Samuelson, P. und Scotchmer, S. (2002), 'The law and economics of reverse engineering', *Yale Law Journal* **111**(7), S. 1575–1663.

Weber, S. (2004), *The success of open source*, Harvard University Press, Cambridge, MA.

Kapitel 4

Recht und Politik

„Erst wenn die Mutigen klug und die Klugen mutig geworden sind, wird das zu spüren sein, was irrtümlicherweise schon oft festgestellt wurde: ein Fortschritt der Menschheit."

– Erich Kästner (1899-1974)

Von Lizenzen und Patenten

KATJA LUTHER

1. Einleitung zum Kapitel Recht und Politik

Bei dem Begriff Open Source denkt man in erster Linie an Softwareentwicklung und evtl. noch an ein Vertriebskonzept für Software, aber wahrscheinlich nicht an Recht und Politik. Doch vor allem im letzten Jahr konnte man mehrfach erfahren, welchen Einfluss Entscheidungen der Politik oder Urteile von Gerichten haben können.

In diesem Kapitel wird dieser Einfluss aus der Sichtweise von Juristen (Thomas Ebinger und Andreas Neumann) sowie eines Journalisten (Stefan Krempl) betrachtet.

2. Softwarepatente – all überall

> „Ein Start-up mit keinen eigenen Patenten wird künftig gezwungen sein zu zahlen, was die Größen zu verlangen belieben und der Preis könnte hoch sein – etablierte Firmen haben ein Interesse daran, künftige Wettbewerber auszugrenzen.“[1]

Durch die Debatte über Softwarepatente wurde im letzten Jahr in breiterer Öffentlichkeit als zuvor auch über Softwarelizenzen und Open Source diskutiert. Man erkennt die Brisanz dieses Themas auch an dem Anteil, den es in diesem Kapitel hat. Bisher fiel Software unter das Urheberrecht, wodurch zwar der Quelltext an sich geschützt wurde, nicht aber die Idee, die dahinter steckte. Die Implementierung eines alternativen Programms zur Bearbeitung eines Problems, das bereits durch ein ähnliches Programm gelöst wurde, verletzt kein gültiges Recht. Das Urheberrecht schützt also die konkrete Ausdrucksform einer Lösung, Patente hingegen eher die Lösungsidee an sich.

Patente wurden als Anreiz zum Schaffen neuer Erfindungen eingeführt. Nach der Erteilung des Patentes muss die Erfindung in der Patentschrift veröffentlicht werden, der Patentinhaber erhält aber das absolute Recht an der Erfindung. In der Softwareentwicklung ist es jedoch zu bezweifeln, dass die Patentierung eines Programmes

1 Bill Gates zitiert in Sietmann (2002)

oder eines Algorithmusses zu einem Innovationsanreiz führt. Zum einen ist die Entwicklung in der Softwareproduktion so schnell, dass eine patentbedingte Blockierung eines Algorithmusses oder eines Computerprogrammes eher zu einer Bremse der Innovation führt bzw. große Firmen stark bevorzugt. Nur sie sind in der Lage, die kostenintensive Patentrecherche und die anfallenden Patentlizenzen zu bezahlen. Zum anderen liegt es nahe, dass die freie Weiterentwicklung existierender Lösungen auch eher zu Innovationsförderung führt, anstatt dass jeder das Rad neu erfinden muss.

Bei der Vergabe von Patenten gelten feste Regeln, wann einer Erfindung ein Patent zusteht:

Anhand der Beschreibung muss ein Fachmann in der Lage sein, die Erfindung nachzubauen, die Erfindung muss neu sein und eine gewisse Erfindungshöhe aufweisen, das heißt, dass nicht jeder andere bei der Lösung desselben Problems zur gleichen Lösung kommen würde. Außerdem muss die Erfindung einen wirtschaftlichen Nutzen haben. Patente sichern dem Erfinder für eine gewisse Zeit die Rechte an seiner Erfindung.

In Europa wurde durch das Europäische Patentübereinkommen (EPÜ) 1973 das Patentrecht einheitlich geregelt. Es wurde auch definiert, welche Erfindungen nicht durch Patente geschützt werden können. In Artikel 52(2) EPÜ werden „Programme für Datenverarbeitungsanlagen" ausdrücklich von der Patentierbarkeit ausgenommen. Schon 1985 wurde diese Formulierung aufgeweicht, indem man statt der Software als solche nur noch nicht-technische Software von der Patentierbarkeit ausschloss. Nun war es an den Patentrichtern zu entscheiden, wann ein Programm technischer Natur war und wann nicht. Diese Entscheidung konnte natürlich sehr unterschiedlich ausfallen. Durch diese rechtliche Lücke wurden in den letzten Jahren in Europa schon etwa 30 000 Softwarepatente erteilt.[2]

Um wieder eine einheitliche Linie in das europäische Patentrecht zu bekommen, entwarf die Kommission der Europäischen Union 2002 eine neue Richtlinie zu „computerimplementierten Erfindungen". Diese wurde dann zunächst vom Europäischen Parlament diskutiert. Am 24. September 2003 legte das Parlament seinen Standpunkt fest, in dem der Schwerpunkt wieder stärker auf die „Technizität" einer Erfindung gelegt wurde. Dadurch wurde eine stärkere Einschränkung der Patentierbarkeit von Software gefordert als in der Richtlinie der Kommission.

Der Ministerrat der EU wiederum machte 2004 ebenfalls einen Vorschlag für die Richtlinie über Patentierbarkeit „computerimplementierter Erfindungen". Dieser geht wieder in die Richtung der Richtlinie der Kommission und erlaubt nahezu jegliche Patentierung von Software. Die Richtlinie des Rates führte bei vielen Interessensverbänden zu Protest. Nicht nur die Open-Source-Gemeinde fürchtet, nicht mehr ohne Patentverletzungen programmieren zu können, auch kleine und mittelständische Unternehmen fürchten um ihre Existenz. Sie argumentieren, dass nur große Softwareunternehmen von dieser Regelung profitieren werden, da sie zum einen häufig schon jetzt große Mengen an Patenten besitzen, die sie unentgeltlich nutzen können und zum anderen haben sie die finanziellen Mittel, um bei der Entwicklung für die

2 http://swpat.ffii.org/patente/zahlen/

nötige Patentrecherche und evtl. anfallende Patentgebühren aufzukommen. Da die EU einem Monopol oder Oligopol entgegenwirken sollte, kann die Verabschiedung der Richtlinie des EU-Rates eigentlich nicht im Interesse der EU sein.

Im Dezember 2004 sollte der Rat über diese Richtlinie abstimmen, doch durch das Votum Polens, das noch Beratungsbedarf sah, wurde die Verabschiedung noch einmal verschoben. In der Bundesrepublik Deutschland verabschiedete der Bundestag Ende Januar 2005 einen Antrag, der sich ausdrücklich für eine „effektive" Begrenzung der Patentierbarkeit von Software aussprach und die Justizministerin Zypries darauf hinwies, dies auch im EU-Rat durchzusetzen (Krempl 2005*a*). Obwohl die Bundesregierung anfangs noch den Standpunkt des Parlaments teilte, unterstützte sie bis zum Votum Polens die Richtlinie des Rates.

Der letzte Stand bis zum Redaktionsschluss war, dass das EU-Parlament eine Rekonsultation der Debatte über die Patentierbarkeit „computerimplementierter Erfindungen" bei der EU-Kommission beantragte, der weitere Verlauf jedoch noch nicht geklärt wurde (Krempl 2005*b*). In dem Artikel von Stefan Krempl wird die Entwicklung der Richtlinie zu Softwarepatenten genauer betrachtet und auch die Auswirkungen auf die Softwarebranche analysiert.

3. GPL – rechtskräftig

Neben der Diskussion um Softwarepatente gab es im letzten Jahr eine wichtige Entscheidung des Münchner Landesgerichts zu der Rechtsgültigkeit der Open-Source–Lizenz *General Public Licence* (GPL). Einleitend zunächst ein paar Worte zu Open–Source-Lizenzen allgemein. Der Open-Source-Ansatz steht den Befürwortern von Softwarepatenten genau entgegen. Die Open-Source-Bewegung ist davon überzeugt, dass gerade Software, deren Quelltext offen und frei verfügbar ist, Innovation fördert und dazu motiviert, neue Lösungen zu finden. Die weitere Verwertung von Open–Source-Software wird mittels zugehöriger Lizenzen geregelt. Diese Open-Source–Lizenzen legen fest, welche Rechte, aber auch welche Pflichten der Verwerter der Software besitzt. Eine Lizenz wird als Open-Source-Lizenz anerkannt, wenn sie die von der Open-Source-Initiative geforderten Richtlinien erfüllt.[3]

Trotz dieser Richtlinien gibt es noch sehr unterschiedliche Lizenzen. Zum Beispiel im Hinblick auf die Weiterverwertung der Software oder von Teilen des Quellcodes sehen die Lizenzen verschiedene Pflichten für den Nutzer vor.

Die beiden bekanntesten Lizenzen sind die GPL und die BSD (Berkeley Software Distribution). Außer bezüglich der Richlinien, die von Open-Source-Lizenzen gefordert werden, unterscheiden sich diese beiden Lizenzen sehr stark. Die am weitesten verbreitete Open-Source-Lizenz ist wahrscheinlich die GPL, die 1989 von Richard Stallman und Eben Moglen ins Leben gerufen wurde. Software, die unter der GPL[4] veröffentlicht wird, darf ein Nutzer nur weiterverbreiten, wenn dies auch wieder unter den Bedinungen der GPL geschieht. Das Prinzip, dass jede weiterentwickelte Soft-

3 http://www.opensource.org/docs/definition.php
4 http://www.fsf.org/licenses/gpl.html

ware, die auf einer GPL lizenzierten Software basiert, auch wieder unter der GPL lizenziert werden muss, nennt man „Copyleft“[5]. Der Begriff Copyleft wurde von den Entwicklern der GNU-Lizenz geprägt. Ein Programmierer nutzt also zunächst sein „Copyright“(Urheberrecht), um seiner Software Vertriebsbestimmungen hinzuzufügen, die jedem erlauben, das Produkt weiterzuverbreiten und zu verändern. Dann legt der Programmierer mit der „Copyleft“ genannten Voraussetzung fest, dass jede Weiterverbreitung oder Veränderung auch wieder unter der GPL stehen muss.

Der Name der BSD-Lizenz stammt von einer UNIX-Distribution der Berkeley Universität[6], die als Open Source herausgegeben wurde. Der Lizenztext ist sehr kurz[7] und beschränkt die Weiterverbreitung nur dahin gehend, dass die Lizenz beigefügt werden muss. Der wichtigste Unterschied zur GPL ist, dass der Lizenznehmer die Software auch im Binärcode und kommerziell verbreiten kann, es existiert keine Pflicht zur Weitergabe des Quellcodes.

Bisher gab es keine gerichtliche Entscheidung darüber, ob die GPL eine gültige Softwarelizenz ist (Kanzlei Dr. Bahr 2005). Vor dem Münchner Landgericht wurde im März 2004 über die Frage entschieden, ob die GPL in Deutschland gültig ist. In dem Artikel von Thomas Ebinger wird dieser Prozess dokumentiert. Er beschreibt, wie es dazu kam, dass ein deutscher Open-Source-Entwickler gegen eine Firma einen Unterlassungsanspruch geltend machte.

Die angeklagte Firma hatte auf ihrer Webseite Software zum Download angeboten, die ein Modul einer unter der GPL veröffentlichten Software enthielt. Dies war ohne einen Verweis auf die GPL oder einen *Link* auf den Lizenztext geschehen. Das Landgericht München bestätigte nun, dass die GPL als Allgemeine Geschäftsbedingungen gilt und vom Nutzer der Software auch eingehalten werden muss. Was dies bedeutet, hat Thomas Ebinger in seinem Artikel genauer analysiert.

Ein Problem der GPL in Deutschland kann der generelle Haftungsausschluss sein, der aus dem amerikanischen Urheberrecht kommt. Nach deutschem Recht gilt er jedoch nicht. Dadurch kann es in Deutschland dazu kommen, dass keinerlei Haftungsausschluss geregelt ist. Ein Programmierer könnte dann schon bei kleinen Fehlern bzw. für einfache Fahrlässigkeiten haftbar gemacht werden (Krempl 2004, Jaeger 2004). Unter anderem aus diesem Grund wurde in Bremen eine neue Open-Source–Lizenz erstellt, die sich an dieser Stelle von den bisherigen Lizenzen unterscheidet. Sie beschränkt die Haftung und Gewährleistung ausdrücklich auf grobe Fahrlässigkeit und Vorsatz.[8]

4. Die Politik und ihr Verhältnis zu Open Source

Beschäftigt man sich mit Recht und Politik in Bezug auf Open Source, kam man im letzten Jahr an den zwei schon erwähnten Entwicklungen rund um Softwarepatente

5 http://www.gnu.org/copyleft/copyleft.html
6 http://www.berkeley.edu/
7 http://www.opensource.org/licenses/bsd-license.php
8 http://www.osci.de/bibliothek/bremer_lizenz.pdf

und Open-Source-Lizenzen nicht vorbei. Darüber hinaus sollte man die allgemeine Einstellung der Europäischen Union bzw. der deutschen Politik bezüglich Open Source betrachten.

4.1. Europäische Union

Für den europäischen Softwaremarkt, der im Gegensatz zum amerikanischen vor allem aus kleinen und mittleren Betrieben besteht, sind die Ziele der Europäischen Union von ähnlich großer Bedeutung wie die Ziele der einzelnen Länder. Dies resultiert aus der Tatsache, dass Software im Allgemeinen nicht an Ländergrenzen gebunden ist. Aus diesem Grund ist der Blick auf die Ziele der EU in Hinblick auf Open-Source-Software von besonderer Bedeutung. Andreas Neumann analysiert die Ziele und Aufgaben der EU und inwieweit sich diese in der Unterstützung von Open-Source-Projekten widerspiegeln.

Die EU ist ursprünglich als Wirtschaftsunion gegründet worden und auch heute sind die vorrangigen Ziele der Gemeinschaft die Errichtung eines gemeinsamen Marktes und eine Währungs- und Wirtschaftsunion. Da Open-Source-Software in den allermeisten Fällen entgeltfrei zur Verfügung gestellt wird, kommt Neumann zu dem Ergebnis, dass deren Abgabe entgegen dem Konzept der Wirtschafts- und Währungsunion steht. Quelloffene Software tritt in Form kostenfreier Konkurrenzprodukte sozusagen in Konkurrenz zum herkömmlichen Wettbewerb. Allerdings fördert sie auch die vor- und nachgelagerten Märkte (wie den Hardwaremarkt bzw. den Dienstleistungsmarkt). Außerdem arbeitet Andreas Neumann heraus, dass durch die Veröffentlichung von Sourcecode auch Netzeffekte entstehen, die zu einer nicht zu vernachlässigenden Innovationsförderung führen. Doch auch im Hinblick auf Monopole und die Interoperabilität beschäftigt sich die EU mit der Förderung quelloffener Softwareentwicklung. Das Ziel der Europäischen Gemeinschaft ist, eine möglichst hohe Interoperabilität im Bereich der Informations- und Kommunikationstechnologie zu erreichen. Da große Netze einen Wettbewerbsvorteil gegenüber kleineren Netzen haben, birgt die Bildung solcher Netze die Gefahr von Monopolbildung des Netzanbieters. Um dies zu verhindern, versucht die EU eine Sicherstellung von Interoperabilität zu erreichen – unter anderem durch Berücksichtigung von Open-Source-Software. Die Förderungspolitik der EU steht im Gegensatz zu der Initiative des Ministerrates, die Richtlinie über „computerimplementierte Erfindungen" zu verabschieden (siehe oben). In dem sogenannten „Aktionsplan *e*Europe" wird die Förderung von Open–Source-Software vor allem in den Bereichen *e*Learning, *e*Security und *e*Government gefordert.

4.2. Bundesrepublik Deutschland

> „Wir bekennen uns zu offenen Standards und zur Vielfalt in der Software-Landschaft der Behörden, weil durch den Wettbewerb um beste

Lösungen Qualität und Innovationen gefördert werden."[9]

Auch in Deutschland gehen bereits einige Kommunen und Verwaltungen den Weg in Richtung Open Source. Das sicherlich bekannteste Beispiel ist die Stadt München mit dem LiMux-Projekt[10], das im letzten Sommer kurz wegen Unsicherheiten bezüglich der eventuell entstehenden Kosten durch Patentgebühren ausgesetzt, nach wenigen Tagen jedoch wieder aufgenommen wurde (Löding 2004). Die Bundesregierung hat ähnlich wie die EU einen Aktionsplan zur Förderung von *e*Government-Lösungen: „Bund Online 2005". Es ist geplant, bis 2005 alle internetfähigen Dienstleistungen auch im Internet anzubieten, wobei Innenminister Otto Schily darauf hinweist, dass die Bundesregierung auf Vielfalt im Bereich der Software achten wolle (Bundesministerium des Innern 2003).

Die Projektgruppe von „Bund Online 2005" arbeitet eng mit der KBSt (Koordinierungs- und Beratungsstelle für Informationstechnik in der Bundesverwaltung) zusammen. Durch Bereitstellung von Informationen zu Migrationen und Open–Source-Projekten fördert die KBSt die Verwendung von Open-Source-Software in der Bundesverwaltung.[11] Für Verwaltungen, die vor der Entscheidung stehen, ob sie ihre Software auf Open Source umstellen wollen, wurde der Migrationsleitfaden herausgegeben, der vor allem aus wirtschaftlichen Gesichtspunkten Hilfestellung bei der Entscheidung bietet. Es wurden also gute Voraussetzungen geschaffen für eine möglichst breite Software-Vielfalt in der öffentlichen Verwaltung und somit auch für den Einsatz von Open-Source-Software. In den nächsten Jahren wird sich zeigen, ob die Chancen genutzt werden. Auch im Bereich der Softwarepatente hat die Bundesregierung große Verantwortung und sollte sich dieser in der Diskussion über die kommende Richtlinie über „computerimplementierte Erfindungen" auch bewusst sein. Eine Unterstützung der bisherigen Richtlinie würde eine Förderung der Monopole weniger großer Softwareentwickler nach sich ziehen.

Literaturverzeichnis

Bundesministerium des Innern (2003), 'Schily: Bundesregierung entwickelt Migrationsleitfaden für Open-Source-Software', *http://www.staat-modern.de/* . http://www.staat-modern.de/E-Government/Pressemitteilungen-,10113.555145/Schily-Bundesregierung-entwick.htm?global.printview=2 [07. Feb 2005].

Jaeger, T. (2004), 'Ohne Verantwortung?', *http://www.ifross.de* . http://www.ifross.de/ifross_html/art3.html [07. Feb 2005].

Kanzlei Dr. Bahr (2005), 'LG München I: GPL ist wirksam', *http://www.heyms-drbahr.de/news/* . http://www.heyms-drbahr.de/news/news_det_20040724111658.html [07. Feb 2005].

Krempl, S. (2004), 'Erste Open-Source-Lizenz made for Germany', *heise newsticker* . http://www.heise.de/newsticker/meldung/45764 [07. Feb 2005].

9 Parlamentarische Staatssekretärin im Bundesinnenministerium Ute Vogt auf dem Linux-Tag 2004 (Bundesministerium des Innern 2003)

10 http://www.muenchen.de/Rathaus/dir/limux/projekt/89257/index.html

11 http://www.kbst.bund.de/Software/-,74/Open-Source.htm

Krempl, S. (2005*a*), 'Rechtsausschuss des Bundestags stimmt gegen Softwarepatente', *heise newsticker* . http://www.heise.de/newsticker/meldung/55568 [07. Feb 2005].

Krempl, S. (2005*b*), 'Softwarepatentrichtlinie: EU-Parlament verlangt Neustart des Verfahrens', *heise newsticker* . http://www.heise.de/newsticker/meldung/55878 [07. Feb 2005].

Löding, T. (2004), 'München legt Linux-Projekt wegen der Softwarepatente auf Eis', *heise newsticker* . http://www.heise.de/newsticker/meldung/49735 [07. Feb 2005].

Sietmann, R. (2002), 'Wissen ist Geld', *c't magazin für computertechnik* **24**, S. 108. http://www.heise.de/ct/02/24/108/ [07. Feb 2005].

Quelloffene Software auf Ebene der Europäischen Gemeinschaft

ANDREAS NEUMANN[1]

Software ist seit jeher ein kaum an nationale Grenzen gebundenes Gut, die nationalstaatliche Steuerungsfähigkeit auf diesem Gebiet entsprechend eingeschränkt. Umso bedeutsamer sind für die betreffenden Märkte und Akteure die Ziele und Initiativen inter- und supranationaler Organisationen wie insbesondere auch der Europäischen Gemeinschaft (EG). Der Beitrag schildert die – bislang noch im Wesentlichen fragmentarischen – Ansätze der EG mit Bezug auf quelloffene Software (Open-Source-Software) und die gegenwärtige Tätigkeit der EG auf diesem Gebiet. Dabei dienen die Maßahmen zur Interoperabilitätsfänderung und die Richtlinie zur Patentierbarkeit computerimplementierter Erfindungen (Softwarepatente) als Referenzpunkte auf Mikroebene vor dem Hintergrund der Makroebene des Primärrechts.

1. Einleitung

Die fortschreitende europäische Integration hat eine Verlagerung politischer Entscheidungsprozesse von der nationalen auf die Gemeinschaftsebene bewirkt. Dieser Befund ist zwar alles andere als neu, aber von zentraler Bedeutung für das Verständnis realpolitischer Entscheidungsprozesse.

Auf der Ebene der Mitgliedstaaten wird zunehmend nur noch umgesetzt, was zuvor auf der Ebene der Europäischen Gemeinschaft beschlossen wurde. Damit ist nicht unbedingt ein Bedeutungsverlust nationaler Politiksetzungsfähigkeit, aber jedenfalls eine Verschiebung der Einflussnahmemöglichkeiten zwischen den einzelnen mitgliedstaatlichen Staatsgewalten verbunden. Das zentrale Gesetzgebungsorgan der Gemeinschaft, dessen Zusammensetzung unmittelbar die politischen Machtverhältnisse auf mitgliedstaatlicher Ebene widerspiegelt, ist natürlich der Rat – und dieser setzt sich aus Vertretern der Mitgliedstaaten auf Ministerebene zusammen.[2] Der mitgliedstaatliche Gesetzgeber wird also nicht mehr nur rein faktisch (aufgrund der überlegenen Ressourcen der Ministerialbürokratie), sondern – über den Umweg über Europa – auch rechtlich im Wesentlichen durch die mitgliedstaatliche Exekutive vorgesteuert.

1 Der Verfasser dankt Frau *Julia Wetzel* für wertvolle Vorarbeiten zu diesem Beitrag.

2 Art. 203 Abs. 1 EG-Vertrag.

Diese Tendenzen müssen zwar (zu Recht) demokratietheoretische Bedenken wecken. Sie scheinen aber hierzulande dem insoweit der Verantwortung zu eigenständigen Entscheidungen enthobenen mitgliedstaatlichen Gesetzgeber sogar durchaus gelegen zu sein. Dies zeigte jüngst beispielsweise das Gesetzgebungsverfahren zum neuen Telekommunikationsgesetz deutlich, in dem der Deutsche Bundestag auch auf Druck der Kommission der Europäischen Gemeinschaften ohne (gemeinschafts-) rechtliche Not die eigene Steuerungsfähigkeit beschränkte und sich statt der noch seitens der Bundesregierung vorgesehenen sinnvollen Formulierung eigener regulierungspolitischer Zielsetzungen mit der bloßen Übernahme untergesetzlicher Empfehlungskriterien der Kommission begnügte.[3] In jedem Fall verdeutlichen die geschilderten Entwicklungen, dass die nationale Steuerungsfähigkeit auf der Gemeinschaftsebene rechtlich wie politisch überformt ist. Für praktisch alle Politikbereiche jenseits des aus staatstheoretischen Gründen für die Mitgliedstaaten reservierten Bereiches (*domaine réservé*) ist der „Blick nach Brüssel" bzw. in die anderen Kapitalen der Europäischen Gemeinschaft daher nicht nur lohnend, sondern unverzichtbar. Dies gilt damit auch für den Gegenstand des vorliegenden Sammelwerkes, also für die Entwicklung im Bereich der quelloffenen Software. Ausgangspunkt soll dabei auf der Makroebene eine Analyse der Kompatibilität des Konzepts quelloffener Software mit den grundlegenden Zielen und Aufgaben der Europäischen Gemeinschaft sein. Danach werden dann auf der Mikroebene die beiden Politikbereiche dargestellt, in denen quelloffener Software die bislang bedeutendste Rolle auf Gemeinschaftsebene zukommt.

2. Die Makroebene: quelloffene Software als Herausforderung für das EG-Primärrecht

Jenseits des aktuellen Verfassungsgebungsprozesses verfügt die Europäische Gemeinschaft schon seit jeher über eine „Verfassung" im rechtstechnischen Sinne – den (seitdem mehrfach geänderten) Gründungsvertrag aus dem Jahre 1957. Er „verfasst" die Gemeinschaft. Jedes Handeln ihrer Organe muss letzten Endes in ihm seine Legitimationsgrundlage finden, Maßnahmen der Gemeinschaft, die gegen den Vertrag verstoßen, sind rechtswidrig. Dieser Vorrang des EG-Vertrages auch vor anderen Rechtsakten auf Gemeinschaftsebene findet seinen Ausdruck in der Bezeichnung „Primärrecht". Darunter versteht man generell die Gründungsverträge der Europäischen Gemeinschaft (EG), der (praktisch nur noch wenig bedeutsamen) Europäischen Atomgemeinschaft (EAG) und der mittlerweile aufgelösten Europäischen Gemeinschaft für Kohle und Stahl (EGKS), jeweils einschließlich etwaiger Anlagen, Anhänge und Protokolle sowie späterer Ergänzungen und Änderungen. (Koenig und Haratsch 2003, S. 3) Die von Organen dieser Gemeinschaften erlassenen Rechtsakte bezeichnet man demgegenüber als „Sekundärrecht" (Koenig und Haratsch 2003, S. 3) bzw. entsprechend dem oben dargestellten Legitimationszusammenhang als „abgeleitetes

3 Zur Diskussion um das damit in Bezug genommene Regulierungsziel des „funktionsfähigen Wettbewerbs" und der Zentralnorm des § 10 Abs. 2 des neuen Telekommunikationsgesetzes ausführlich Loetz und Neumann (2003).

Recht". In welchen Bereichen und mit welcher Zielsetzung die Gemeinschaft, womit hier und im Folgenden nur noch die EG gemeint sein soll, tätig wird bzw. überhaupt nur tätig werden darf, richtet sich daher letzten Endes nach der Reichweite des EG-Primärrechts.

Die damit auch für nachrangige Maßnahmen maßgebliche Aufgabe der EG wird in Art. 2 des EG-Vertrags definiert, wobei zugleich die Mittel zur Erfüllung dieser Aufgabe festgelegt werden. Die Aufgabe ist auf Förderung gerichtet – dies betrifft zum einen allgemeine soziale Ziele, wie z. B. das der „Gleichstellung von Männern und Frauen", „ein hohes Maß an Umweltschutz und Verbesserung der Umweltqualität" sowie zum anderen allgemein „die Hebung der Lebenshaltung und der Lebensqualität". Bereits der Aufgabenkatalog des Art. 2 macht aber deutlich, dass die Europäische Gemeinschaft (ursprünglich, aber eben auch noch nach der Neubezeichnung der „Europäischen Wirtschaftsgemeinschaft" [EWG] als „Europäische Gemeinschaft" im Jahre 1993) primär eine Wirtschaftsgemeinschaft ist. So gehört zu den in Art. 2 ausdrücklich genannten Aufgaben der Gemeinschaft nämlich gerade auch, die „Entwicklung des Wirtschaftslebens", „ein hohes Beschäftigungsniveau", „ein beständiges, nichtinflationäres Wachstum" sowie „einen hohen Grad von Wettbewerbsfähigkeit und Konvergenz der Wirtschaftsleistungen" zu fördern.

Diese Priorität des Wirtschaftlichen wird auch in der ebenfalls in Art. 2 enthaltenen Festschreibung der Mittel deutlich, die der Gemeinschaft zur Erreichung ihrer Ziele zur Verfügung stehen: Neben den in Bezug genommenen gemeinsamen Politiken und Maßnahmen nach Art. 3 und 4 des EG-Vertrags wird die Gemeinschaft nämlich in erster Linie „durch die Errichtung eines Gemeinsamen Marktes und einer Wirtschafts- und Währungsunion" tätig. Damit wird auch auf instrumentaler Ebene der Gedanke einer Wirtschaftsgemeinschaft betont. Dabei nimmt der EG-Vertrag nicht Bezug auf eine beliebige Wirtschaftsform, sondern bekennt sich – mit Ausnahmen für bestimmte Bereiche – ausdrücklich zur Marktwirtschaft (Vgl. Koenig und Haratsch 2003, S. 3, Rn. 8). Hinzu kommt die Nennung des Gemeinsamen Marktes als besonders hervorgehobenes Mittel zur Erreichung der Ziele der Gemeinschaft.[4] Der EG-Vertrag nimmt also den Markt als ein fundamentales Konzept der ökonomischen Theorie in Bezug. Unter einem Markt versteht man denjenigen Ort, an dem Angebot und Nachfrage nach einem bestimmten Gut, also nach einer Ware oder Dienstleistung, aufeinander treffen (Koenig et al. 2004, S. 37). Zentraler Koordinationsparameter der Güterallokation ist dabei der Preis, so dass das Konzept des Marktes letztlich auf der Idee des entgeltlichen Güteraustausches beruht.[5]

Daraus wird unmittelbar ersichtlich, dass quelloffene Software nur schwer in den konzeptionellen Ansatz einzupassen ist, auf dem die wirtschaftspolitische Fundierung der Gemeinschaft aufbaut. Denn quelloffene Software wird von ihren Produzenten –

4 Siehe dazu EuGH, Slg. 1982, 1409, 1431 f., Tz. 33; Ukrow in: Calliess und Ruffert (2002), Art. 2 EG-Vertrag Rn. 26 und 30.

5 Vgl. auch den expliziten Hinweis auf die Entgeltlichkeit der in Rede stehenden Betätigung als Wesensmerkmal für die Zurechnung zum „Wirtschaftsleben" bei EuGH, Slg. 1974, 1405, 1418, Tz. 4/10; Slg. 1976, 1333, 1340, Tz. 12/13; Slg. 1988, 6159, 6173, Tz. 12; Ukrow in: Calliess und Ruffert (2002), Art. 2 EG-Vertrag Rn. 14.

zwar nicht notwendigerweise, aber doch in der ganz überwiegenden Zahl der Fälle – unentgeltlich (frei) zur Verfügung gestellt.[6] Der (in Geld messbare[7]) Preis spielt als Koordinationsparameter keine Rolle, der Mechanismus des Marktes im herkömmlichen Sinne ist außer Kraft gesetzt. Auf derartige, im unmittelbaren Bezug kostenlose Güter ist der Marktansatz des Primärrechts nicht zugeschnitten. Wo es schon keinen Markt (im auf Austausch gerichteten Sinne des EG-Primärrechts) geben kann, kann es erst recht keinen Gemeinsamen Markt geben. Freie quelloffene Software passt nicht in das Konzept einer (Markt-) Wirtschafts- und Währungsunion.[8]

Allerdings hat natürlich auch quelloffene Software (markt-) wirtschaftliche Auswirkungen. So ist sie einerseits in der Lage, eine ansonsten durch entgeltliche Güter gedeckte Nachfrage zu befriedigen: Erfolgreiche Open-Source-Projekte, wie das Betriebssystem Linux, der Web-Browser Firefox oder die Bürobedarfslösung OpenOffice haben bewiesen, dass sie in der Lage sind, kommerzielle Software mit (ungefähr) entsprechender Funktionalität zu ersetzen – damit verringert quelloffene Software das Volumen der (auf den Austausch von Gütern und Geld gerichteten) jeweiligen Märkte. Auf diese Weise dient sie nicht dem Auf-, sondern gewissermaßen dem Abbau des (gemeinsamen) Marktes und läuft insoweit dem ökonomischen Ansatz des EG-Primärrechts diametral zuwider.

Andererseits ist freie quelloffene Software aber auch Gegenstand von Austauschverhältnissen und somit in marktwirtschaftliche Zusammenhänge im Sinne des EG–Primärrechts eingebunden.[9] Freilich sind diese Zusammenhänge bei quelloffener Software erheblich subtiler als bei herkömmlichen Gütern, da sie zwischen ihrem Anbieter und dem Nachfrager gerade nicht entgeltlich gehandelt wird. Stattdessen erfolgen (markt-) wirtschaftliche Austauschverhältnisse, die durch quelloffene Software gefördert werden, auf Ebenen, die der Stufe des Angebots vor- und nachgelagert sind. So gehen zunehmend Unternehmen der Software- und Hardwareindustrie dazu über, bestimmte Programme – zum Beispiel Treibersoftware – der Allgemeinheit (und insbesondere den Kunden ihrer sonstigen Produkte) als freie quelloffene Software anzubieten.[10] Bereits im Jahr 2001 gaben ca. 13 % aller (befragten) Unternehmen, deren Hauptzweck die Entwicklung von Software ist, an, zumindest gelegentlich quelloffene Software anzubieten (vgl. Blind et al. 2001, S. III). Diese quelloffene Software wird also von bezahlten Programmierern entwickelt (Koenig und Neumann (2004*b*, S. 130),

6 Etwaige Distributionskosten sind kein Entgelt für die Güterproduktion und sollen daher im Folgenden insoweit außer Betracht bleiben.

7 Sicherlich ist es möglich, auch immaterielle Entlohnungen wie Anerkennung und Prestigegewinn als Engelte im weiteren Sinne zu verstehen, vgl. etwa Ghosh (1998). Ein solches Verständnis wirtschaftlicher Handlungen würde aber jedes menschliche Handeln, soweit es der Erreichung bestimmter Ziele dient, erfassen; es würde auf diese Weise konturenlos werden und eher menschliches Verhalten allgemein als spezifisch die Mechanismen beschreiben, die der Güterverteilung dienen, vgl. auch (Koenig und Neumann 2004*b*, S. 130, Fn. 184).

8 Aus diesem Grund erweist sich auch das EG-Wettbewerbsrecht (Art. 81 ff. EG-Vertrag) als wenig geeignet zur Anwendung auf Sachverhalte mit Bezug zu quelloffener Software, vgl. Koenig und Neumann (2003).

9 Bei entgeltlicher quelloffener Software bestehen keine grundsätzlichen Unterschiede zu anderen Handelsgütern. Auch ein quelloffenes Handelsgut ist ein Handelsgut.

10 Siehe auch Wiebe (2003, S. 163).

Wiebe (2003, S. 163)). Nicht ihr Angebot, aber ihre Produktion ist in diesen Fällen folglich eine typische (markt-) wirtschaftliche Tätigkeit. Auch die Distribution freier quelloffener Software hat (markt-) wirtschaftliches Potential bewiesen,[11] wie namentlich das Aufkommen zahlreicher Linux-Distributionen (und -Distributoren) gezeigt hat. Ähnliche Effekte sind auf den dem eigentlichen Angebot nachgelagerten Ebenen zu verzeichnen: Mit zunehmender Verbreitung und Nutzung freier quelloffener Software sind zugleich der Bedarf und das Angebot an entsprechenden – in erheblichen Teilen durchaus entgeltlichen – Servicedienstleistungen gewachsen (Koenig und Neumann 2004*b*, S. 130; Wiebe 2003, S. 163), wozu auch entsprechende Fachliteratur (Zeitschriften und Bücher) zählt. Hinzu kommen schließlich die effizienz- und innovationserhöhenden Auswirkungen, die von freier quelloffener Software ausgehen können: Durch die Existenz eines kaum mehr überschaubaren Angebots an kostenlosen Programmen besteht die Möglichkeit, ohne allzu große Transaktionskosten verschiedene Softwarelösungen für einen bestimmten Bedarf umfassend zu testen, zu vergleichen und gegebenenfalls auch parallel zu nutzen. Auch Pfadabhängigkeiten bei der Nutzung von proprietären Lösungen können durch freie quelloffene Software vermieden werden. Zugleich führt der Austausch von Programmcode bei quelloffener Software zu einer Aktivierung von Netzeffekten bei der Softwareentwicklung. Die dadurch erzielbaren Innovationseffekte erlauben ihrerseits die Schaffung neuer Programmlösungen (Blind et al. 2001, S. III) und fördern auf diese Weise die Innovationsdynamik (Blind et al. 2001, S. VII) und damit letzten Endes auch die Güterproduktion.

Quelloffene Software ist somit eine Herausforderung für das EG-Primärrecht, wobei dieser Befund zwei unterschiedliche Seiten hat: Auf der einen Seite passt freie quelloffene Software nicht in das dem ökonomischen Ansatz des EG-Vertrages zugrunde liegende marktwirtschaftliche Konzept, das auf der Prämisse unmittelbarer Austauschverhältnisse beruht. Stattdessen kann sie derartige Austauschverhältnisse sogar in solch einer Weise beeinflussen, dass der Güteraustausch auf den betroffenen Märkten zurückgeht. Freie quelloffene Software ist insoweit „Wettbewerber des Wettbewerbs". Auf der anderen Seite kann sie den Wettbewerb auf solchen Ebenen fördern, die dem eigentlichen Produktangebot vor- und nachgelagert sind, insbesondere auf Dienstleistungsmärkten.[12] überdies kann sie die gesamtwirtschaftliche Effizienz und Innovation fördern, was ebenfalls wettbewerbsstimulierend wirkt. Die Analyse des EG-Primärrechts ergibt damit ein disparates Bild. Hieraus erklärt sich, dass die Gemeinschaft bislang noch keine übergeordnete Strategie mit Blick auf (freie) quelloffene Software entwickelt hat (bzw. überhaupt entwickeln konnte), wie es sie

11 Vgl. auch Schulz (2004, S. 573–574).

12 Auch in der Stellungnahme des Wirtschafts- und Sozialausschusses zu der „Mitteilung der Kommission an den Rat und das Europäische Parlament, den Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen: Sicherheit der Netze und Informationen: Vorschlag für einen europäischen Politikansatz" (2002/C 48/07), ABl. EG 2002 C 48, 33, 35, wird konstatiert, dass „sich ein wichtiger Wirtschaftssektor von Diensten für Unternehmen" um das Konzept freier quelloffener Software „entwickelt (habe), der von einigen Grossunternehmen des Informatiksektors getragen wird." Der Ausschuss sieht in freier quelloffener Software „eine Form gesunder Konkurrenz zu den monopolistischen Tendenzen des Softwaremarktes sowie des sich ständig weiterentwickelnden Marktes an Netzdiensten".

in anderen Bereichen – etwa der Telekommunikation oder des Eisenbahnwesens – gibt. Quelloffene Software spielt daher in der Realität der EG primär auf einzelnen Mikroebenen eine Rolle.

3. Die erste Mikroebene: quelloffene Software als Beitrag zur Sicherstellung von Interoperabilität

Ein Wirtschaftsbereich, der in der Politik der Gemeinschaft in den letzten fnfzehn Jahren eine zunehmend prominente Rolle eingenommen hat, ist die Branche der Informations- und (Tele-) Kommunikationstechnologien. Seit 1990 hat die EG konsequent und zielstrebig die Öffnung des zuvor in fast allen europäischen Ländern staatsmonopolistisch organisierten Telekommunikationssektors betrieben.[13] Bis zum 1. Januar 1998 mussten die Mitgliedstaaten (grundsätzlich) den Betrieb von Telekommunikationsnetzen und das Angebot von Telekommunikationsdiensten vollständig liberalisiert haben, so dass der Telekommunikationssektor auf allen Ebenen dem Wettbewerb geöffnet war. Dieser Wettbewerb wurde zugleich durch ein umfassendes System aus Netzzugangsansprchen und flankierender Regulierung gefördert, das nach einer grundlegenden Reform im Jahr 2002 in konsolidierter und ausdifferenzierter Form,[14] auf bislang nicht absehbare Zeit fortbestehen wird. Diese Liberalisierung der Telekommunikationsmärkte traf auf eine technische Entwicklung von radikaler Dynamik, in deren Folge Informations- und Kommunikationstechnologien, vor allem auch auf Grundlage des Internets, so gut wie alle Lebensbereiche erfasst und durchdrungen haben. Die dadurch entstandenen Märkte sind praktisch ausnahmslos durch Netzcharakteristika geprägt. Netzeffekte – also insbesondere der (grundsätzliche) Nutzenzuwachs, der für alle Teilnehmer aus dem Anschluss eines zusätzlichen Teilnehmers folgt – führen aber grundsätzlich zu einem natürlichen Wettbewerbsvorteil großer Netze (Koenig et al. 2004, S. 42). Diese ökonomische Besonderheit birgt die wettbewerbspolitische Gefahr, dass es zur Bildung enger Oligopole oder sogar zur Monopolstellung eines einzelnen Netzes und damit zu nicht wettbewerblich kontrollierten Machtpositionen der betreffenden Anbieter kommt. Doch auch wenn eine solche Entwicklung vermieden werden kann, begründen Netzeffekte jedenfalls potentielle Pfadabhängigkeiten, die zu gesamtwirtschaftlich nicht wünschenswerten Ergebnissen führen können (Koenig und Neumann 2004*b*, S. 116). Ein zentrales Ziel, das die Gemeinschaft im Bereich der Informations- und Kommunikationstechnologien verfolgt, ist daher die Sicherstellung von Interoperabilität.[15] Wenn (Netz-) Systeme mit anderen (Netz-) Systemen zusammenarbeiten können, sind Netzeffekte erzielbar, ohne dass das Gesamtsystem unter der Kontrolle nur eines Betreibers stehen muss.

13 Ausführlich hierzu und mit umfangreichen Nachweisen zu den jeweiligen Maßnahmen (Koenig et al. 2004, S. 57 ff.).

14 Siehe hierzu Koenig et al. (2004, S. 73 ff.).

15 Siehe ausdrücklich zum Stellenwert von Interoperabilitätsregulierung die Mitteilung der Kommission über Hemmnisse für den breiten Zugang zu neuen Diensten und Anwendungen der Informationsgesellschaft durch offene Plattformen beim digitalen Fernsehen und beim Mobilfunk der dritten Generation, KOM (2003) 410, S. 8.

Zugleich kann es in Teilsystemen zu technischer Innovation kommen, ohne dass dabei auf die Vorteile eines Rückgriffs auf die installierte Basis der anderen Teilsysteme verzichtet werden msste.[16]

Dabei hat die EG die besondere Eignung von quelloffener Software zur Sicherstellung von Interoperabilität erkannt. Zwar erfordert Interoperabilität im Wesentlichen nur die Offenheit von *Schnittstellenspezifikationen* und nicht auch der weiteren Netzkomponenten (Blind et al. 2001, S. III). Sind jedoch die Komponenten selbst offen, sind es zwangsläufig auch die Schnittstellen.[17] Gerade angesichts der Erfahrungen mit der nicht immer ausgeprägten Neigung von Softwareherstellern, die Schnittstellen ihrer Programme vollständig und auf wettbewerbsneutrale Weise offen zu legen,[18] kommt der Verbreitung quelloffener Software – und ihrer Förderung – insoweit ein besonderer Wert zu. Da die reaktive Herstellung der Offenheit von Schnittstelleninformationen mit den Mitteln des allgemeinen Wettbewerbsrechts zeitintensiv und auch darüber hinaus mit erheblichen praktischen Schwächen belastet ist (Koenig et al. 2003, S. 404 ff.), gilt dies gerade in den dynamischen Softwaremärkten und benachbarten Märkten der Information und Kommunikation in noch gesteigertem Maß. Die EG erkennt diesen Nutzen quelloffener Software an und operationalisiert sie in geeigneten Teilbereichen zur Erreichung des generellen Interoperabilitätsziels. Da die Verfügbarkeit quelloffener Software aber von dem Eigenengagement ihrer Hersteller abhängig ist und dieses zumindest in der Mehrzahl der Fälle nicht von wirtschaftlichen (oder gar rechtlichen) Überlegungen geleitet ist, sind die direkten Steuerungsmöglichkeiten der Gemeinschaft insoweit sehr reduziert. Sie muss sich daher im Wesentlichen auf eine Förderung quelloffener Software als Bestandteil ihrer Interoperabilitätsstrategie beschränken.

So begründete die Kommission der Europäischen Gemeinschaften in ihrer Mitteilung zur Organisation und Verwaltung des Internets schon im Jahr 2000 ihre Absicht „sicherzustellen, dass die bestehende Neutralität der Internet-Spezifikationen in Bezug auf Betriebssysteme und Hardware-Plattformen beibehalten und weiterentwickelt wird", primär mit dem „zunehmenden Interesse [. . .] der Nutzer an offener Software".[19] Die Gemeinschaft setzt aber vor allem mit ihrer Förderungspolitik explizit positive Anreize, verstärkt quelloffene Software zu entwickeln und einzusetzen. Diese Anreizsetzungen sind in einem breiteren Kontext der von der Kommission vorbereiteten und vom Europäischen Rat im Jahr 2000 beschlossenen Strategie für einen beschleunigten Übergang zu einer wettbewerbsfähigen und dynamischen wissensgestützten Wirtschaft, dem so genannten „Aktionsplan *e*Europe", zu sehen. Die

16 Siehe zum Ganzen auch ausführlich Koenig et al. (2003, S. 410 ff.).

17 Dieser Erst-recht-Zusammenhang wird auch in der Mitteilung der Kommission an den Rat, das Europäische Parlament, den Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen "Sicherheit der Netze und Informationen: Vorschlag für einen europäischen Politikansatz", KOM (2001) 298 endgültig, S. 25, betont.

18 Siehe hierzu die Microsoft-Entscheidung der Kommission, WuW 2004, 673, 676 ff., und dazu Koenig und Neumann (2004*a*, S. 555, 559) m. w. Nachw, sowie allgemein zur Schnittstellproblematik Koenig und Neumann (2004*b*, S. 132 f.) ebenfalls m. w. Nachw.

19 Mitteilung der Kommission an den Rat und das Europäische Parlament „Organisation und Verwaltung des Internets – Internationale und europäische Grundsatzfragen 1998–2000", KOM (2000) 202, S. 12.

Förderung von quelloffener Software war – beschränkt auf einzelne Sachgebiete, namentlich den Bereich sicherer Netze und intelligenter Chipkarten sowie die Initiativen „Europas Jugend ins Digitalzeitalter" und „Regierung am Netz – elektronischer Zugang zu öffentlichen Dienstleistungen", – bereits in dem ursprünglichen Aktionsplan selbst angelegt und als Aufgabe gerade auch der Kommission und der Mitgliedstaaten definiert.[20] Dieser Konzentration auf die drei – hier der Einfachheit halber im Einklang mit einer fragwürdigen Praxis über englische Kunstbegriffe bezeichneten[21] – Schwerpunktbereiche *e*Security, *e*Learning und *e*Government entspricht die nachfolgende Praxis der Gemeinschaftsorgane:

Bereits im Jahr 2001 forderte der Rat im Zusammenhang mit der Integration von Informations- und Kommunikationstechnologien in die Bildungs- und Ausbildungssysteme (*e*Learning) die Kommission auf, „die Entwicklung von ... 'Open-Source-Software' zu fördern".[22] Im selben Jahr empfahl die Kommission inInteroperabilität ihrer Mitteilung über die Sicherheit der Netze und Informationen überdies die Verwendung quelloffener Software zur Sicherstellung der Interoperabilität von sicherheitsfördernden Lösungen – etwa in Bezug auf elektronische Signaturen –, die überdies „zur schnelleren Fehlerkorrektur sowie zu größerer Transparenz beitragen" könne.[23] Und erst im Frühjahr 2004 berücksichtigten das Europäische Parlament und der Rat in ihrem Programm für den Zeitraum 2005–2009 zur interoperablen Erbringung europaweiter elektronischer Behördendienste („eGovernment-Dienste") für europäische öffentliche Verwaltungen, die Organe der Gemeinschaft und andere Stellen sowie europäische Unternehmen und Bürger[24] ausdrücklich quelloffene Software bei der Festlegung der förderungswürdigen Maßnahmen. Die Bereitstellung und Wartung von „Werkzeuge(n), die auf quelloffener Software basieren", werden dabei als Infrastrukturdienst identifiziert.[25] Als solcher kommen sie für die Durchführung als (von der Gemeinschaft mitfinanzierte) horizontale Maßnahmen im Sinne des Programms in Betracht.[26] Darüber hinaus wird auch die Förderung der Verbreitung bewährter Verfahren bei der Nutzung von quelloffener Software durch öffentliche Verwaltungen als flankierende Maßnahme ebenfalls in den Anwendungsbereich des Förderungsprogramms einbezogen.[27]

Der *e*Europe-Aktionsplan wurde im Jahr 2002 zum „Aktionsplan *e*Europe 2005"[28]

20 Kommission der Europäischen Gemeinschaften, *e*Europe 2002: Eine Informationsgesellschaft für alle – Aktionsplan vorbereitet von Rat und Europäischer Kommission zur Vorlage auf der Tagung des Europäischen Rates am 19./20. Juni 2000 in Feira, 2000, S. 11, 13 und 22.

21 Siehe generell zu fremdsprachigen Einflüssen auf die deutsche Sprache Stickel (2001, S. 159).

22 Entschließung des Rates vom 13. Juli 2001 zum *e*-Learning (2001/C 204/02), ABl. EG 2001 C 204, 3, Entschließungspunkt 10 viii).

23 Kommission, KOM (2002) 202, S. 25.

24 Beschluss 2004/387/EG des Europäischen Parlaments und des Rates vom 21. April 2004 über die interoperable Erbringung europaweiter elektronischer Behördendienste (eGovernment-Dienste) für öffentliche Verwaltungen, Unternehmen und Bürger (IDABC) – in der berichtigten Fassung –, ABl. EG 2004 L 181, 25.

25 Anhang II B. lit. r des Beschlusses 2004/387/EG.

26 Vgl. Art. 5, 6 und 10 des Beschlusses 2004/387/EG.

27 Anhang II C. 3. lit. c des Beschlusses 2004/387/EG.

28 Kommission der Europäischen Gemeinschaften, Mitteilung der Kommission an das Europäische Parla-

hin fortentwickelt. Dabei wurde das Ziel, quelloffene Software – gerade in den genannten drei Bereichen *e*Security, *e*Learning und *e*Government – zu fördern, fortgeschrieben[29] und nunmehr zum Teil deutlich in den übergeordneten funktionalen Zusammenhang der Sicherstellung von Interoperabilität gerückt.[30] Weitere Maßnahmen zur Unterstützung und Ergänzung des Aktionsplans *e*Europe 2005 wurden daraufhin auf Ebene der Forschungsförderung unter dem Themenbereich „Technologien für die Informationsgesellschaft" (TIS)[31] vorgesehen, die ihrerseits auch ausdrücklich quelloffene Software berücksichtigten. Erneut wurde die Förderung „von Software mit frei zugänglichem Quellcode" als wichtige Maßnahme definiert, „wenn dies erforderlich ist, um die Interoperabilität der Lösungen sicherzustellen und Innovationen voranzubringen".[32] Für elektronische Behördendienste wurde hier sogar eine generelle Priorität für den Einsatz quelloffener Software definiert.[33] Auch für den Bereich elektronischer Gesundheitsdienste (*e*Health) wurde insoweit nunmehr grundsätzlich eine Förderung der Verwendung quelloffener Software in Aussicht gestellt.[34] Außerdem wurde erneut mehrfach auf den Interoperabilitätsnutzen verwiesen, der durch den Einsatz quelloffener Software – etwa im Bereich der Softwareentwicklung und der Forschungstätigkeit – erzielt werden kann.[35]

4. Die zweite Mikroebene: Softwarepatente als Bedrohung für quelloffene Software

Auf einer zweiten Mikroebene, auf der quelloffener Software in der Politik der Europäischen Gemeinschaft besondere Bedeutung zukommt, geht es weniger um die Frage, ob und wofür quelloffene Software nützlich ist. Stattdessen geht es bei der nunmehr bereits seit einigen Jahren sehr engagiert geführten Debatte über Softwarepatente – und ihre explizite normative Harmonisierung durch die Gemeinschaft – um die Frage, welche Auswirkungen die eventuell an nicht sonderlich hohe Voraus-

ment, den Rat, den Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen „*e*Europe 2005: Eine Informationsgesellschaft für alle – Aktionsplan zur Vorlage im Hinblick auf den Europäischen Rat von Sevilla am 21./22. Juni 2002", KOM (2002) 263 endg.

29 Kommission der Europäischen Gemeinschaften, Aktionsplan *e*Europe 2005, KOM (2002) 263 endg., S. 12 und 18. Lediglich im Bereich des *e*Learnings fand quelloffene Software keine ausdrückliche Erwähnung mehr auf Ebene des Aktionsplans selbst.

30 Vgl. insbesondere Kommission der Europäischen Gemeinschaften, Aktionsplan *e*Europe 2005, KOM (2002) 263 endg., S. 12 („Bis Ende 2003 wird die Kommission einen abgestimmten Rahmen für die Interoperabilität bekannt geben, der die Bereitstellung europaweiter elektronischer Behördendienste für Bürger und Unternehmen unterstützen soll. [. . .] Grundlage werden offene Normen sein, und die Verwendung von Software mit frei zugänglichem Quellcode wird unterstützt.").

31 Kommission der Europäischen Gemeinschaften, Technologien für die Informationsgesellschaft: Ein vorrangiger Themenbereich für Forschung und Entwicklung im Rahmen des Spezifischen Programms „Integration und Stärkung des Europäischen Forschungsraums" des 6. Rahmenprogramms der Gemeinschaft – Arbeitsprogramm 2003 – 2004.

32 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 7.

33 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 20 („Sie sollten . . . soweit wie möglich Softwarelösungen mit frei zugänglichem Quellcode wählen.").

34 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 22.

35 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 26 und 38.

setzungen geknüpfte Möglichkeit eines Patentschutzes für Rechnerprogramme auf quelloffene Software hätte und ob sich diese Konsequenzen angesichts der Vorteile von Softwarepatenten rechtfertigen lassen. Diskutiert wird die Frage anlässlich des Gesetzgebungsverfahrens zu einer Richtlinie über die Patentierbarkeit computerimplementierter Erfindungen. Mit dieser Richtlinie soll vor allem die Anwendung des Patentrechts bei der Erteilung von Softwarepatenten in Rechtsprechung und Praxis der Mitgliedstaaten vereinheitlicht werden, um so Hemmnisse für den Binnenmarkt abzubauen. Schon bevor die Kommission im Jahr 2002 ihren initialen Richtlinienvorschlag[36] vorlegte, hatten sich Entwickler und Anwender von quelloffener Software als Hauptgegner einer ausdrücklichen Regelung (bzw. grundsätzlichen Bestätigung) von Softwarepatenten zu erkennen gegeben.[37]

Die Diskussion um Softwarepatente führt im Kern auf den grundlegenden Widerspruch gewerblicher Schutzrechte zurück, die der Sache nach hoheitlich gesicherte Monopolrechte sind, von denen eine wettbewerbsfördernde Wirkung ausgehen soll. Dahinter steht die Überlegung, dass die Aussicht auf Monopolgewinne Unternehmer und Erfinder zu Innovationen antreibt.[38] Zum Teil wird sogar noch weiter gehend zwischen „Gütern", „Produktion" und „Innovation" als drei Wettbewerbsebenen unterschieden und davon ausgegangen, dass Wettbewerb auf einer Ebene eine Beschränkung des Zugangs auf der darunter liegenden Ebene erforderlich macht.[39] Danach setzt Wettbewerb auf Ebene der Produktion Eigentum an Gütern, also den Ausschluss des freien Zugangs zu bestehenden Gütern voraus. Und Wettbewerb auf der Innovationsebene würde wiederum die Unterdrückung von Wettbewerb auf der Produktionsebene erfordern, was gerade durch gewerbliche Schutzrechte wie Patente oder das Urheberrecht erreicht würde.[40] Auch wenn die Hypothese, dass die Aussicht auf Marktmacht Treiber für Innovationen ist, bislang empirisch nicht erhärtet wurde, ist sie plausibel und in einem gewissen Umfang sicherlich auch zutreffend (Koenig et al. 2002, S. 28). Auch bei den Softwarepatenten ist die eigentliche Frage daher diejenige nach diesem Umfang.

Indem Patente funktionale Aspekte schützen, erfasst ein Softwarepatent unabhängig von der konkreten Implementierung jedes Programm, das einen Rechner zu einem patentgeschützten Verhalten veranlasst (Gehring und Lutterbeck 2003, S. 6; Wiebe 2003, S. 163). Da Softwarepatente einerseits verhältnismäßig preiswert zu produzieren sind und andererseits bei einem Programmablauf regelmäßig eine Vielzahl von – potentiell patentgeschützten – Funktionen erfüllt werden, hat die Zulassung von

36 Kommission der Europäischen Gemeinschaften, Vorschlag für eine Richtlinie des Europäischen Parlaments und des Rates über die Patentierbarkeit computerimplementierter Erfindungen, KOM (2002) 92 endgültig.

37 Vgl. Kommission der Europäischen Gemeinschaften, KOM (2002) 92 endgültig, S. 3.

38 Daneben wird oftmals darauf verwiesen, dass die Monopolgewinne zur (Re-) Finanzierung kostspieliger Innovationen auch benötigt werden. Dieses Argument ist offensichtlich nicht generell gültig, sondern auf Branchen mit entsprechenden Kostenstrukturen beschränkt. Für die Pharmaindustrie dürfte es daher eher Gültigkeit beanspruchen als – zumindest bei genereller Heranziehung – für die Softwarebranche.

39 von Weizsäcker (1980, passim)

40 Siehe hierzu auch Koenig et al. (2002, S. 54).

Softwarepatenten zur Folge, dass sich Softwareproduzenten einem „Patentdickicht“[41] gegenübersehen und Softwareentwicklung mit (zumindest potentiell) hohen Lizenz- bzw. Ausweichkosten behaftet wird.[42] Für die Anbieter kommerzieller Software müssen insoweit wechselseitige Lizenzen (*Cross Licensing*) und die Einrichtung so genannter Technologiepools[43] praktische Problemlösungsstrategien darstellen (Gehring und Lutterbeck 2003, S. 16 f.) oder letzten Endes eine Abwälzung der Kosten auf den Käufer in Betracht kommen. Diese Möglichkeiten haben die Entwickler (freier) quelloffener Software hingegen in der Regel nicht(Gehring und Lutterbeck 2003, S. 18). Softwarepatente können für quelloffene Software daher tendenziell existenzbedrohende Auswirkungen haben.[44]

Bislang ist noch nicht absehbar, wie dieser grundlegende Interessenkonflikt auf europäischer Ebene letzten Endes entschieden werden wird. Das Europäische Parlament hatte in der ersten Lesung des Richtlinienentwurfs nicht nur eine besondere Beobachtung der Auswirkungen computerimplementierter Erfindungen auf die Open-Source-Bewegung vorgeschrieben. Viel weitgehender hatte es vor allem die Anforderungen an patentwürdige Software gegenüber dem ursprünglichen Vorschlag der Kommission erheblich angehoben und insbesondere eine Ausnahmeregelung für die Verwendung patentierter Technologien zum Zwecke der Interoperabilität vorgesehen.[45] Der Rat legte sich hingegen in einer am 18. Mai 2004 erzielten politischen Einigung auf eine wieder stärker dem Kommissionsvorschlag angenäherte Fassung fest, in der die Einschränkungen, die das Europäische Parlament vorgesehen hatte, wieder weitgehend zurückgenommen waren.[46] Von mitgliedstaatlicher Ebene werden jedoch seitdem zunehmend Bedenken gegenüber dem im Rat gefundenen Kompromiss laut – nach einem Beschluss des niederländischen Parlaments, in welchem die niederländische Regierung aufgefordert wird, der Einigung im Rat die Unterstützung zu entziehen, liegen nunmehr auch ähnlich ausgerichtete Entschließugsanträge der

41 Zum Begriff siehe Shapiro (2000, passim.).

42 Siehe zum Ganzen etwa Gehring und Lutterbeck (2003, S. 6 ff.), m. w. Nachw.

43 Siehe zu diesem Modell Kommission der Europäischen Gemeinschaften, Leitlinien zur Anwendung von Artikel 81 EG-Vertrag auf Technologietransfer-Vereinbarungen, ABl. EG 2004 C 101, 2, Rn. 210 ff., und dazu (Koenig und Neumann 2004*a*, S. 555,558 f.).

44 Dementsprechend wird eine Ausweitung der Patentierbarkeit von Software durch Entwickler quelloffener Software generell negativ bewertet und auch von anderen (kommerziellen) Softwareproduzenten Einschränkungen bei der Entwicklung (freier) quelloffener Software als Folge einer (weiteren) Ausweitung der Patentierbarkeit erwartet Blind et al. (2001, S. V f.). Keine ernsthafte Gefährdung für quelloffene Software erwartet hingegen Wiebe (2003, S. 163), der jedoch von „weiteren erheblichen Hürden für eine Patentierung“ ausgeht, die in der Realität zurzeit freilich zusehends abgebaut werden.

45 Bericht über den Vorschlag für eine Richtlinie des Europäischen Parlaments und des Rates über die Patentierbarkeit computerimplementierter Erfindungen, A5-0238/2003 endgültig; Standpunkt des Europäischen Parlaments festgelegt in erster Lesung am 24. September 2003 im Hinblick auf den Erlass der Richtlinie 2003/.../EG des Europäischen Parlaments und des Rates über die Patentierbarkeit computerimplementierter Erfindungen, ABl. EG 2004 C 77 E, 230. Siehe hierzu auch Hemler (2004), EuZW 2004, 257, unter Hinweis auf die Notwendigkeit einer Vereinheitlichung des Softwareschutzes in der Rechtspraxis der USA und der EG.

46 Rat der Europäischen Union, Politische Einigung über den gemeinsamen Standpunkt des Rates, Dok. 9713/04.

Fraktionen des Deutschen Bundestages vor.[47] Es ist somit noch nicht einmal gesichert, dass auf die politische Einigung die (momentan für Ende November 2004 vorgesehene) formelle Verabschiedung eines gleichlautenden gemeinsamen Standpunktes folgt. Selbst wenn der Rat aber bei seiner Linie bleibt, ist derzeit nicht prognostizierbar, wie sich das Europäische Parlament in der dann anstehenden zweiten Lesung zu dieser Richtlinienentwurfsfassung positionieren wird. Dies gilt in besonderem Maß, da das Parlament aufgrund der zwischenzeitlich erfolgten Neuwahlen neu zusammengesetzt ist und somit noch nicht einmal von einer politischen Kontinuität ausgegangen werden kann.

5. Fazit

Die EG hat noch keine einheitliche Haltung zu quelloffener Software gefunden. Die Idee des (regelmäßig) unentgeltlichen Angebots von Gütern lässt sich nicht ohne Friktionen in den Grundansatz einer auf den unmittelbaren Leistungsaustausch gerichteten Wirtschaftsgemeinschaft integrieren, um die es sich bei der EG im Kern nach wie vor handelt. Eine Analyse der betroffenen Mikroebenen spiegelt diesen grundsätzlichen Zielkonflikt, der auf der Makroebene des Primärrechts besteht, deutlich wider: Einerseits erkennt die Gemeinschaft die Schnittstellenfunktion von quelloffener Software als Plattform zur Herstellung von Interoperabilität mit all ihren wettbewerbsfördernden Wirkungen insbesondere auf benachbarten Märkten an. Andererseits nehmen zumindest die Kommission und der Rat zur Förderung des Wettbewerbs auf den Softwaremärkten eine durchaus (zumindest potentiell) existentielle Bedrohung quelloffener Software in Kauf, indem sie trotz der zusehends ausufernden Anwendungspraxis Softwarepatente ausdrücklich normativ anerkennen wollen. Zugleich zeigt der Streit um die Softwarepatente die Schwierigkeiten auf, die wettbewerbs- und wohlfahrtsfördernden Wirkungen quelloffener Software in das überkommene System der europäischen Wettbewerbspolitik einzupassen. Die auf Mikroebene festgestellten Widersprüche verdeutlichen überdies auch, dass die Gemeinschaft zumindest zurzeit noch keine übergeordnete Strategie in Bezug auf quelloffene Software verfolgt. Das ist Risiko und Chance zugleich: Das Risiko liegt darin, dass auf den einzelnen Mikroebenen nicht aufeinander abgestimmte Strategien verfolgt werden und es auf diese Weise zu Widersprüchlichkeiten und Fehlentwicklungen kommt. Die Chance besteht darin, dass die Mitgliedstaaten noch in der Lage sind, einen künftigen übergeordneten Ansatz der Gemeinschaft mit Blick auf quelloffene Software proaktiv mitzugestalten. Sie sollten diese Chance nutzen.

47 Vgl. die Anträge der Fraktion der FDP, BT-Drs. 15/3240, der Fraktion der CDU/CSU, BT–Drs. 15/3941, sowie der Fraktionen der SPD und von Bündnis 90/Die Grünen, BT-Drs. 15/4034.

Literaturverzeichnis

Blind, K., Edler, J., Nack, R. und Straus, J. (2001), 'Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen: Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation (Kurzfassung)', Karlsruhe.

Calliess, C. und Ruffert, M. (Hrsg.) (2002), *Kommentar zu EU-Vertrag und EG-Vertrag*, 2. Aufl., Hermann Luchterhand Verlag, Neuwied/Kriftel.

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/file/.

Ghosh, R. A. (1998), 'Cooking pot markets: an economic model for the free trade in free goods and services on the Internet', *First Monday* **3**(3).

Hemler, T. (2004), 'Softwarepatente – quo vadis?', *Europäische Zeitschrift für Wirtschaftsrecht* S. 257.

Koenig, C. und Haratsch, A. (2003), *Europarecht*, Mohr Lehrbuch, 4. Aufl., J. C. B. Mohr (Paul Siebeck), Tübingen.

Koenig, C., Loetz, S. und Neumann, A. (2003), Innovation im Spannungsverhältnis von Markt und Regulierung, *in* D. Klumpp, H. Kubicek und A. Roßnagel (Hrsg.), 'Next generation information society?', Talheimer Verlag, Mössingen-Talheim, S. 403–412.

Koenig, C., Loetz, S. und Neumann, A. (2004), *Telekommunikationsrecht*, Betriebs-Berater Studium, UTB/Verlag Recht und Wirtschaft, Heidelberg.

Koenig, C. und Neumann, A. (2003), 'Standardisierung und EG-Wettbewerbsrecht – ist bei vertrauenswürdigen Systemumgebungen wettbewerbspolitisches Misstrauen angebracht?', *Wirtschaft und Wettbewerb* S. 1138–1152.

Koenig, C. und Neumann, A. (2004*a*), 'Neue wettbewerbspolitische und -rechtliche Entwicklungen zum „Trusted Computing"', *Datenschutz und Datensicherheit* S. 555–560.

Koenig, C. und Neumann, A. (2004*b*), Wettbewerbsrechtliche Aspekte vertrauenswürdiger Systemumgebungen, *in* C. Koenig, A. Neumann und T. Katzschmann (Hrsg.), 'Trusted Computing', Schriftenreihe Kommunikation & Recht, Verlag Recht und Wirtschaft, Heidelberg, S. 100–140.

Koenig, C., Vogelsang, I., Kühling, J., Loetz, S. und Neumann, A. (2002), *Funktionsfähiger Wettbewerb auf den Telekommunikationsmärkten*, Schriftenreihe Kommunikation & Recht, Verlag Recht und Wirtschaft, Heidelberg.

Loetz, S. und Neumann, A. (2003), 'The Scope of Sector-specific Regulation in the European Regulatory Framework for Electronic Communications', *German Law Journal* S. 1307–1334.

Schulz, C. (2004), 'Open Source Software vor Gericht', *Multimedia und Recht* S. 573–574.

Shapiro, C. (2000), 'Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard-Setting', Working Paper No. CPC00-11, Berkeley.

Stickel, G. (2001), Das „Fremdwort" hat ausgedient, *in* A. Koch und A. Neumann (Hrsg.), 'Ut desint vires, tamen est laudanda voluntas – Festschrift für Christian Celsen zum Bestehen des ersten juristischen Staatsexamens', Edition Octopus, Verlagshaus Monsenstein und Vannerdat, Münster, S. 159–170.

Wiebe, A. (2003), 'Open Source Software – eine erfolgreiche Alternative', *Telekommunikations- & Medienrecht* S. 163.

von Weizsäcker, C. C. (1980), *Barriers to Entry: A Theoretical Treatment*, Springer-Verlag, Berlin/Heidelberg/New York.

Der Kampf gegen Softwarepatente – Open Source im Auge des Sturms

STEFAN KREMPL

Am Brüsseler Richtlinienprojekt über die „Patentierbarkeit computerimplementierter Erfindungen" scheiden sich die Geister: Konzerne in der Computer- und Telekommunikationsindustrie erhoffen sich davon eine Harmonisierung der Rechtssituation innerhalb der EU und eine Sanktionierung der bereits recht weitgehenden Praxis des Europäischen Patentamtes. Überraschend starke Lobbygruppen aus dem Mittelstand warnen dagegen vor einer Flut an „Trivialpatenten" nach US-Muster und schwerwiegenden Auswirkungen auf den Wettbewerb allgemein sowie auf die Entwicklung freier Software im Besonderen. Das Gesetzgebungsverfahren gestaltet sich daher als rechtlicher Krimi mit Klassenkämpfen, Ränkespielen und zahlreichen Verzögerungen. Auch im dritten Jahr nach der Veröffentlichung des ersten Richtlinienvorschlags zeichnet sich noch kein Kompromiss zwischen Kommission, Rat und Parlament ab. Der Beitrag gibt einen Überblick über die wichtigsten Frontlinien.

1. Einleitung

Kaum eine Frage wird im Bereich des so genannten „geistigen Eigentums" momentan stärker diskutiert als die der Gesetzgebung über die Patentierbarkeit von Computercode. Anlass ist die von der Europäischen Union angestrebte EU-weite „Harmonisierung" in diesem Bereich. Den formalen Prozess hat die EU-Kommission Anfang 2002 mit ihrem Vorschlag für eine Richtlinie zur „Patentierbarkeit computerimplementierter Erfindungen"[1] gestartet. Nach langem Ringen und einer heftigen Lobbyschlacht, an der sich erstmals neben der Großindustrie und ihren Branchenverbänden wie der EICTA (European Information, Communications and Consumer Electronics Technology Industry Associations) auch Organisationen wie der Förderverein für eine Freie Informationelle Infrastruktur (FFII) oder die Free Software Foundation Europe (FSFE) und die durch sie vertretenen mittelständischen Betriebe und Software-Entwickler beteiligten, folgten im September 2003 die Änderungen des

1 http://europa.eu.int/comm/internal_market/en/indprop/com02-92de.pdf

Europaparlaments. [2] Der EU-Rat, in dem die nationalen Regierungen vertreten sind, konterte im wahrsten Sinne des Wortes nach einem längeren Tauziehen in den eigenen Reihen mit der „politischen Einigung" über seinen Standpunkt im Mai 2004.[3] Die polnische Regierung verzögerte die offizielle Verabschiedung der Ratsposition allerdings wiederholt. Sie brachte Bedenken vor, dass der Text des Ministergremiums die Patentierbarkeit von Computercode nicht ausschließen würde. Gleichzeitig beschäftigte sich das EU-Parlament Anfang 2005 mit einem Antrag zum kompletten Neustart des Gesetzgebungsverfahrens. Ein Ende des Streits oder ein Kompromissvorschlag waren zur Drucklegung dieses Bandes nicht in Sicht.

Open Source steht im Zentrum der hitzigen Debatte um Softwarepatente, die sich rund um die Brüsseler Direktive entwickelt hat. Zwar betonen vor allem die Gegner eines ausgeweiteten Monopolschutzes für Computercode jenseits des Urheberrechts zurecht immer wieder, dass es keineswegs um freie Software allein geht. Von Trivialpatenten im Softwarebereich sind Entwickler proprietärer, also auf das klassische Modell „geistigen Eigentums" setzende Computerprogramme, letzten Endes genauso betroffen wie die Open-Source-Welt. Qua definitionem oft sehr breit ausgerichtete Softwarepatente legen hauptsächlich kleinen und mittelständischen Entwicklerbetrieben Steine in den Weg. Denn diese können sich im Gegensatz zu Konzernen die nicht unbedeutenden Kosten des Patentsystems nicht leisten. Weder verfügen sie über Patentanwälte, die gezielt die entwickelte Software auf patentierbare Algorithmen und Erfindungen hin abklopfen, noch können sie in der Regel die Gebühren für die Anmeldung und Aufrechterhaltung zahlreicher Patente auf sich nehmen. Der Mittelstand bleibt damit weitgehend außen vor im Spiel der Patentanwälte mit ihren Patentportfolios, in dem die Großen untereinander teuren Verletzungsklagen mit Kreuzlizenzierungsverträgen zuvorkommen. Aufgrund des komplexen Motivationsgefüges, das als Umfeld für die Entstehung von Open-Source-Software von Nöten ist (Osterloh et al. 2004), ist die Welt des frei verfüg- und modifizierbaren Quellcodes allerdings im Kern von Softwarepatenten gefährdet. Die Ursache liegt im Copyleft[4], der wesentlichen Stellschraube des Systems freie Software. Dieser auf Richard Stallman, den Doyen der *Free Software Foundation*, zurückgehende Mechanismus sorgt dafür, dass die derart lizenzierte Software frei verfügbar bleibt. Das Copyleft sichert das öffentliche, nicht das private Eigentumsrecht. Es baut somit auf dem Copyright auf, „pervertiert" es aber letztlich und führt den Gedanken einer Monopolverwertung eines Computerprogrammes ad absurdum. Dieses Prinzip kann mit dem Urheberrecht als dessen „Parasit" noch gut leben. Es beißt sich aber mit dem Patentrecht, da es dessen „Schutzgedanken" vollständig aufbricht und die monopolartige Verwertung einer in Software gegossenen Erfindung ja gerade verhindert. Andererseits steht und fällt die Entwicklung freier Software mit dem die Rechte der Allgemeinheit stärkenden Copyleft. Falls es wegbrechen oder zugunsten des ihm entgegengesetzten Paradigmas des Patentschutzes auf Computercode aufgegeben werden sollte, würde das gesamte

2 http://www2.europarl.eu.int/omk/sipade2?SAME_LEVEL=1&LEVEL=3&NAV=X&DETAIL=&PUBREF=-//EP//TEXT+TA+P5-TA-2003-0402+0+DOC+XML+V0//DE

3 http://register.consilium.eu.int/pdf/de/04/st09/st09713.de04.pdf

4 http://www.gnu.org/copyleft/copyleft.html

System der Anreize zur Produktion freier Software in sich zusammenfallen.

Dazu kommt noch ein ganz pragmatischer Grund, der Open-Source-Software theoretisch in besonderer Weise „anfällig" macht für Patentstreitigkeiten: Da der Quellcode offen liegt und für jeden Interessierten einsehbar ist, lassen sich Programmabschnitte, die Patente verletzen könnten, leichter herauspicken. Daher führen Firmen wie Microsoft umgekehrt als eines ihrer Argumente für Softwarepatente den Trend ins Feld, dass sie im Rahmen von Lizenzprogrammen wie „Shared Source" selbst immer mehr proprietären Quellcode für bestimmte Nutzergruppen wie Regierungen offen legen würden und einen gesonderten Patentschutz für ihre nun nicht mehr in geheimen Programmzeilen versteckten Erfindungen bräuchten.[5] Allerdings lässt diese Darlegung außer Acht, dass schon das Urheberrecht derlei befürchtete Plagiate unterbinden würde.

2. Die Münchner LiMux-Migration und die Angst vor Patentklagen

Dass Softwarepatente für die mittelständische Wirtschaft sowie die öffentliche Hand, die verstärkt ihre Rettung vor der Redmonder Lizenzschraube und den Microsoftschen Monopolgefährdungen in der Flucht in Open Source sieht, zur Bedrohung werden, ist im Sommer 2004 vor allem mit dem Theater um den kurzfristigen Stopp der Linux-Migration der Stadt München ins Zentrum der allgemeinen Aufmerksamkeit gerückt. Weil es sich beim LiMux-Projekt[6], in dessen Rahmen rund 14 000 Desktop-Rechner und Server komplett auf freie Software umgestellt werden sollen, um eine weltweit Beachtung findende Unternehmung handelt, war das Medienecho groß, als die bayerischen Pioniere Anfang August plötzlich die Ausschreibung des Kernelements „LiMux Basis-Client" mit der Begründung stoppten, dass mit der Softwarepatentlinie des EU-Rats „rechtliche und finanzielle Risiken" im großen Maßstab auf die Stadtverwaltung zukommen könnten. Hatten sich zuvor vor allem Fachmagazine intensiv mit der problematischen Haltung des Ministerrats auseinandergesetzt (Krempl 2004*a*), berichtete nun gar der Spiegel über den drohenden Spießrutenlauf der freien Software-Entwickler über die Bajonette der Brüsseler Patentklauseln (Evers und Traufetter 2004).

Anlass für die Alarmstimmung im Münchner Rathaus waren zwei Anträge der Grünen im Stadtrat. Sie forderten Oberbürgermeister Christian Ude nachdrücklich auf, bei der Bundesregierung in Sachen Softwarepatente zu intervenieren. Sollte das Bundesjustizministerium seine befürwortende Haltung zur EU-Richtlinie über die „Patentierbarkeit computerimplementierter Erfindungen" in der Version des Ministerrates nicht ändern, sah der Hauptantragsteller, der grüne Stadtrat Jens Mühlhaus, „unkalkulierbaren Schaden" auf München zukommen. Durch die vorgesehene breite Patentierbarkeit von Software könnten hohe finanzielle Forderungen für die Nutzung patentierter Verfahren entstehen oder der Einsatz entsprechender Programme blo-

5 Marie Therese Huppertz, zitiert in Krempl (2004*b*)

6 http://www.muenchen.de/Rathaus/referate/dir/limux/89256/index.html

ckiert werden. Dies sei nicht nur verheerend für die mittelständischen Firmen, mit denen die Münchner die Linux-Migration hauptsächlich schultern wollen, sondern könne zum Ausfall kompletter Referate in der Verwaltung führen.

Die Kurzschlusshandlung der Münchner löste heftige Reaktionen aus, die von „albern" bis „Microsoft in die Hand spielend" reichten. Selbst die CSU machte sich plötzlich Sorgen um Softwarepatente, weil sie hoffte, das von ihr strikt abgelehnte LiMux-Projekt über diesen Haken doch noch zum Wanken bringen zu können. Gleichzeitig zeigte die Panik aber auch, wie groß die Verunsicherung durch die Softwarepatentrichtlinie in Europa inzwischen ist. Die *Free Software Foundation Europe* und der LinuxTag e. V. gehen gar von einer Einladung zur „psychologischen Kriegsführung" in der Wirtschaft durch die Brüsseler Bestrebungen aus. „Ein wirres Gerücht ist völlig ausreichend, um ein komplexes und aufwendiges Projekt für Tage aus der Spur zu bringen", konstatiert FSFE-Präsident Georg Greve. Der LinuxTag-Vorsitzende Oliver Zendel sieht darüber hinaus „Mechanismen aus dem kalten Krieg" in der gegenwärtigen Aufrüstung der Patentportfolios bei Konzernen am Werk. Leidtragende seien „Programmierer, klein- und mittelständische Betriebe und somit der Wirtschaftsstandort Europa".

Tatsächlich füllt gerade Microsoft momentan rasant seinen Softwarepatentkorb. Der Softwarehersteller besaß Mitte 2004 rund 4 500 erteilte Patente, etwa 5 000 weitere waren anhängig. Die Softwarepatente schützen unter anderem Methoden, mit denen eine Datei abgespeichert oder Text auf einem Monitor dargestellt wird. Sein Portfolio will der Softwaregigant mit neuen Lizenzpolitiken lukrativ verwerten. Ein weiteres Ziel dürfte die Einschüchterung des Open-Source-Lagers sein. Angriffsfläche bietet Linux, das aus seinen Innereien kein Geheimnis macht, theoretisch allemal. Zumindest hat die US-Firma *Open Source Risk Management* 283 potentielle Patentansprüche entdeckt, die durch das frei verfügbare Betriebssystem verletzt werden könnten (Ravicher 2004). Davon hat aber noch keiner vor Gericht Stand gehalten. Ähnlich nur bedingt stichhaltig ist eine kurzfristige Patentrecherche des Fördervereins für eine Freie Informationelle Infrastruktur. Demnach soll die allein die Desktop–Umrüstung im Rahmen von LiMuX von rund 50 potenziellen Patentverletzungen betroffen sein (Blasum 2004).

Das Bundesjustizministerium hält die Ängste der Softwarepatentgegner für unberechtigt. Im Haus der SPD-Politikerin Brigitte Zypries kann niemand einen Zusammenhang zwischen den LiMux-Querelen und der Brüsseler Direktive erkennen. „Freie Software ist ein wichtiger Innovationsfaktor für den Standort Deutschland", versichern Ministeriumsvertreter immer wieder. Es sei zudem kein Fall in Deutschland bekannt, in dem Open Source in einem Patentverletzungsverfahren angegriffen worden sei. Doch mit dieser Position heizte das Ministerium die Debatte nur weiter an. So stellte der LinuxVerband in einem Schreiben an das Ressort nach der LiMux-Verzögerung klar, dass schon heute „eine Reihe von patentierten Algorithmen beispielsweise in den Bereichen Kryptographie, Multimedia und Datenformate in freier Software nicht genutzt werden können – mit negativen Folgen für Sicherheit, Interoperabilität und Wettbewerb". Die Bundesregierung müsse Farbe bekennen, da sie nicht einerseits einer breiten Patentierbarkeit von Programmen das Wort reden, andererseits Linux

als Softwarelösung nutzen könne.

Die Zuspitzung der Diskussion durch die Münchner LiMux-Pause fand nicht nur Befürworter. „Man kann nur davor warnen, die Problematik 'Softwarepatente' allein auf freie Software zu reduzieren", gab Till Jäger vom Institut für Rechtsfragen der Freien und Open Source Software (ifrOSS) zu bedenken. Herkömmliche Softwareanbieter seien von den Unsicherheiten ebenso betroffen. Die Hysterie um das Münchner Projekt ist nach Ansicht des Juristen fehl am Platz. Wünschenswert wäre eine genaue Prüfung, ob die für LiMux benötigte Software tatsächlich unter ein Benutzungsverbot fällt, um gegebenenfalls fehlerhaft erteilte Patente für nichtig erklären zu lassen. Dies könnte auch ein Licht auf die Punkte werfen, die in Brüssel tatsächlich geregelt werden müssten.

Entsprechende Schritte versprach Ude mit Hilfe eines Rechtsgutachtens einzuleiten. In einem Positionspapier hat die Stadt zudem eine Reihe offener Fragen aufgelistet (Ude 2004), etwa nach der langfristigen Investitionssicherheit der für LiMux veranschlagten rund 30 Millionen Euro, nach der sich abzeichnenden Patentflut im Softwarebereich sowie nach den sich daraus ergebenden Klagewellen und deren Folgen für die Innovationskraft. Mit der „Münchner Linie" fordert der SPD-Politiker „Rechtssicherheit" aus Berlin und klare Formulierungen in der Brüsseler Richtlinie. Unterstützung erhofft er sich auch von anderen Kommunen wie den ebenfalls auf Open Source umrüstenden Schwäbisch Hallern, die von einer Gesetzesänderung auf EU-Ebene negativ betroffen sein könnten. Linux will Ude aber auf jeden Fall die Treue halten. Den Ausschreibungsstopp hob er bereits „nach einer Denkpause von wenigen Tagen" wieder auf und leitete die aktive Migrationsphase ein.

Die von der Stadt München angeforderte Rechtsbeurteilung der Risiken möglicher Patentklagen bei der Linux-Migration im Rahmen des LiMux-Projekts, auf dessen Basis der Stadtrat im September 2004 offiziell grünes Licht gab für die Fortsetzung des Betriebssystem- und Applikationswechsels, schoss aber am Kernthema größtenteils vorbei und stärkte ironischerweise just der Richtlinienversion des Ministerrats den Rücken. So stieß sich der FFII angesichts des Gutachtens der Münchner Kanzlei Frohwitter (Sedlmaier und Gegerich 2004) vor allem daran, dass das 42-seitige Papier in weiten Teilen allein die neuere Auslegung des Europäischen Patentübereinkommens[7] durch den Bundesgerichtshof und die darauf aufbauende, softwarepatentfreundliche Spruchpraxis des Europäischen Patentamts rechtfertigt und erst in den Schlusskapiteln auf die eigentlichen Ängste des Auftraggebers eingeht.

Für die Autoren des Gutachtens, einen Rechts- und einen Patentanwalt, gehört der Monopolschutz für computerimplementierte Erfindungen „seit über 30 Jahren in Europa" zum Rechtsalltag. Damit seien Patentverletzungen durch Computerprogramme „heute bereits möglich", wobei dies aber für proprietäre und Open-Source-Software genauso gelte. Eine konkrete Untersuchung der Patentproblematik bei Linux und des gesamten darauf aufbauenden Münchner Projekts sparten sich die Kanzleimitarbeiter aufgrund dieser Herangehensweise an das Problem. „Sollte die Stadt München durch den Einsatz von Software ein Patent verletzen", führen sie allgemein aus, „wird

7 http://www.european-patent-office.org/legal/epc/d/ma1.html

zumeist ein vergleichsweise geringer Streitwert anzusetzen sein." Patente auf „computerimplementierte Erfindungen" würden sich regelmäßig nicht auf ein Programm als Ganzes beziehen, sondern nur auf einzelne Funktionalitäten. Die Gutachter zeigten sich so zuversichtlich: Letztlich könne jede patentrechtliche Funktionalität mit einer hohen Wahrscheinlichkeit ersetzt beziehungsweise „gegen Entrichtung einer angemessenen Gebühr lizenziert werden". Die Stadt solle aber vorsichtshalber Dokumentationszentren der Open-Source-Gemeinde unterstützen, „die den jeweiligen Stand der Technik in den unterschiedlichsten Bereichen der Softwaretechnik mit einem entsprechenden Zeitstempel versehen" und so bei Patentklagen die Frage der Neuheit von „computerimplementierten Erfindungen" klären helfen könnten.

3. Von der Wirkung der Naturkräfte zu Programmansprüchen

Wodurch unterscheiden sich die im Mittelpunkt der Debatte stehenden Richtlinienversionen im Kern überhaupt? Das Europaparlament hat in 1. Lesung mit einer Reihe von Änderungen am ursprünglichen Richtlinienvorschlag der EU-Kommission reinen Softwarepatenten einen effektiven Riegel vorgeschoben (Wiebe 2004). Der für eine Patenterteilung erforderliche technische Erfindungsbeitrag wird darin eng definiert: Er muss sich auf ein „gewerbliches Anwendungsgebiet" beziehen, „das zur Erreichung vorhersehbarer Ergebnisse der Nutzung kontrollierbarer Kräfte der Natur bedarf." Zudem sollen die EU-Mitgliedstaaten sicherstellen, „dass die Datenverarbeitung nicht als Gebiet der Technik im Sinne des Patentrechts betrachtet wird". In einem weiterführenden Artikel hat das Parlament ein Beispiel angeführt, welche Merkmale eine Patentierbarkeit computerimplementierter Lösungen in diesem Sinne ausschließen sollen: Ermögliche eine Erfindung allein „Einsparungen von Ressourcen innerhalb eines Datenverarbeitungssystems", dürfe der Patentschutz nicht gewährt werden. Letztlich fordern die Abgeordneten, dass eine computerimplementierte Erfindung auch eine „industrielle Anwendung" nach sich ziehen müsse. Um die Verbreitung von Standards zu erleichtern, wollen die Volksvertreter ferner das Umgehen patentierter Techniken zur Erreichung von Interoperabilität zwischen Computersystemen erlauben.

Die vom Ministerrat der EU verabschiedete Version versucht Interoperabilität dagegen nur über das allgemeine Wettbewerbsrecht abzusichern. Eine unzureichende Klausel, befinden die SPD-Bundestagsabgeordneten Ulrich Kelber und Jörg Tauss. Sei in der Computerindustrie doch die Praxis eingerissen, dass an den Schnittstellen zwischen einzelnen Systemen und Lösungen „gerade mithilfe des Patentrechts Marktabschottungsversuche unternommen und Marktzutrittsschwellen für Wettbewerber erhöht werden". Der Rat verzichtet zudem auf jegliche Definition von Technizität. Ferner soll der Umfang zulässiger Patentansprüche laut dem besonders umstrittenen Artikel 5(2) Rechte auf Computerprogramme als solche sowie auf Programme, die auf einem Datenträger vorliegen, umfassen. Schon die Veröffentlichung einer Software auf einem Server könnte damit als Patentverstoß gewertet werden, ohne dass das entsprechende Programm von einem Computer überhaupt ausgeführt würde. Kritiker sehen daraus Möglichkeiten zur Zensur von Software-Entwicklern erwachsen. Kelber und Tauss etwa fürchten, dass diese so genannten Programmansprüche

allein „zu einer Kriminalisierung von Besitztatbeständen und zur Legitimierung von Durchsetzungsmaßnahmen im Internet" führen.

4. Zick-Zack-Kurs der Bundesregierung

Eine unrühmliche Rolle beim Festzurren der Ratsposition hat die Bundesregierung gespielt. So hatte Elmar Hucko, Ministerialdirektor im Justizministerium, auf einer Kundgebung von Softwarepatentgegnern in Berlin im Mai 2004 zunächst heftige Bedenken gegen den Richtlinienvorschlag der irischen Ratspräsidentschaft öffentlich zur Sprache gebracht. Er prangerte unter anderem die Tatsache an, dass Patente verstärkt als Strategie zum „Niederknüppeln der Konkurrenz" missbraucht würden. In der entscheidenden Abstimmung enthielt sich Deutschland allerdings nicht wie von Justizministerin Zypries zuvor angekündigt, sondern verhalf dem Papier nach kosmetischen Veränderungen zum Segen des Ministerrats. Ihr Staatssekretär Hansjörg Geiger hatte dabei nur bewirkt, dass der technische Beitrag einer „computerimplementierten Erfindung" laut dem Ratspapier nun „neu" und erfinderisch sein muss – Anforderungen, die das Europäische Patentamt aber bereits generell stellt. Man habe zwar noch zusätzliche Klarstellungspunkte gesehen, erläutert Ministerialdirigent Raimund Lutz: „Wir wollten die Technizität näher definieren", beteuert der Experte für geistiges Eigentum. „Aber das war im internationalen Bereich nicht vermittelbar." In anderen Ländern habe die Sorge vorgeherrscht, dass in der Richtlinie ein zu statischer Begriff festgeschrieben würde, der später nicht an die technische Entwicklung anzupassen sei.

Noch weniger konsequent präsentierte sich das Bundeswirtschaftsministerium. So startete die Behörde Mitte Juli 2004 eine Unternehmensbefragung zum Streitthema Softwarepatente als Teil einer größeren Studie zur Interoperabilität in der IT-Industrie – damals unter großem Zeitdruck, um die Diskussion um die Richtlinie noch mit Zahlen und Fakten zu unterfüttern. Nachdem der Branchenverband Bitkom die kurzfristig anberaumte Umfrage wegen einer vermeintlich tendenziösen Ausrichtung kritisiert hatte, vollzog die Führungsebene des Ressorts allerdings im Herbst einen ungeordneten Rückzug. So entschuldigte sich der damalige Staatsekretär Alfred Tacke in einem Schreiben an den Lobbyverein, dass die Sondierung „zu Missverständnissen und Irritationen über die grundlegende Position der Bundesregierung zum Richtlinienentwurf über die Patentierbarkeit computerimplementierter Erfindungen geführt hat." Der Versuch, im Rahmen der Studie „die Wechselbeziehungen zwischen Interoperabilität, Patentschutz und Wettbewerb" zu untersuchen, werde wohl misslingen, distanzierte sich Tacke vorab von den zu erwartenden Befunden: „Es ist schon jetzt absehbar, dass die Fragebogenaktion keine Resultate bringen wird, die den Anforderungen an wissenschaftlich fundierte Ergebnisse gerecht werden." Gleichzeitig untersagte das Wirtschaftsministerium dem beauftragten Professorenteam von der Fachhochschule Gelsenkirchen, die Auswertung der rund 1 300 – statt der zunächst erwarteten 100 – eingegangenen Fragebögen zu veröffentlichen. Ende Oktober sprang daher die Kampagne NoSoftwarePatents.com ein, hinter der die Firmen 1&1, MySQL und Red Hat stehen. Sie bat die Teilnehmer der Umfrage, die Stellungnahmen noch einmal an

sie zu mailen, um eine eigene Auswertung zu ermöglichen und die bereits investierte Arbeit der Unternehmen doch noch mit Ergebnissen zu belohnen.

Während die Bundesregierung sich so weiter hinter das Ratskonstrukt stellt und eine kritische Auseinandersetzung in den eigenen Reihen vermeidet, demonstrierte der Bundestag bei einer ersten Plenardebatte zu Softwarepatenten im Oktober 2004 seltene Einigkeit. Die CDU zitierte aus Briefen von SPD-Politikern Passagen voller Kritik am Kurs von Justizministerin Zypries, während die Grünen der FDP bestätigten, dass deren Position ihrer eigenen Beschlusslage entspreche. Es gibt zwar Differenzen im Detail. So unterstützen die FDP und die Grünen pauschal den Richtlinienentwurf des Europaparlaments. Die SPD und die Union sehen dagegen leichten Korrekturbedarf bei der Position ihrer Straßburger Kollegen. Strikt ablehnend stehen alle Fraktionen aber der Richtlinienversion des EU-Rats gegenüber. Sollte sie Gesetz werden, sehen die Parlamentarier Wettbewerb und Innovation in der Informationstechnik in Gefahr sowie kleine und mittelständische Software-Entwickler einschließlich Open-Source–Programmierer am Abgrund existenzbedrohender Patentklagen.

5. Erdbeben im System des „geistigen Eigentums“

Jahre der vertieften Auseinandersetzung über die EU-Richtlinie haben bislang allein zu verhärteten Fronten zwischen Softwarepatentbefürwortern und -gegnern geführt. So prallten etwa bei einem „Dialogangebot“, zu dem das Justizministerium und das Deutsche Patent- und Markenamt Ende Oktober 2004 nach München geladen hatten, hochrangige Vertreter beider Lager aufeinander. Dort trauerte Gert Kolle, Hauptdirektor Internationale Angelegenheiten und Patentrecht beim Europäischen Patentamt, den Zeiten hinterher, in denen es rund um Softwarepatente eine „reine Expertendiskussion“ gegeben habe und die Beschwerdekammern seines Amtes die Richtung bestimmt hätten. Mit seiner Aussage, dass ein „Computerprogramm doch per se etwas Technisches ist“, ließ er jenseits allgemeiner Kriterien wie der Höhe oder der Neuheit einer Erfindung kaum Grenzen für die Patentierbarkeit von Algorithmen zu. Sämtliche Versuche, den Begriff der Technik im Rahmen des Brüsseler Gesetzgebungsverfahrens näher zu definieren, verwies er in den Bereich des Unmöglichen: Urteile des Bundesgerichtshofs, die dabei mit dem „Einsatz beherrschbarer Naturkräfte“ oder „mit einem kausal überschaubaren Ergebnis“ hantiert hätten, seien international „nicht zu verkaufen und höhere Philosophie“. Uwe Schrieck, Leiter IP-Strategie bei Siemens und Patentanwalt, verteidigte ebenfalls die Arbeit des Ministerrats: Sein Hause leiste den „großen Teil der Wertschöpfung durch 'Embedded'-Software. Da müssen wir unsere Forschung und Entwicklung absichern und uns gegen Wettbewerber absetzen,“ die sonst die Ergebnisse direkt übernehmen könnten. Inakzeptabel an der Parlamentsversion sei in dieser Hinsicht, dass darin eine „industrielle“ Anwendbarkeit einer Erfindung verlangt werde. Dies sei gerade bei Verfahren im Mobilfunkbereich nicht zu leisten.

Allgemein fürchten Konzerne und Anwaltskammern, dass die vom EU-Parlament vorgesehenen Änderungen Erfindungen zu programmtechnisch eingerichteten Steuerungen in wesentlichen technischen Bereichen betreffen. Als Beispiele nennen sie die

Kraftfahrzeugtechnik, die Stormversorgung, die Bildgenerierung für medizinische Anwendungen, die Telekommunikation oder die Umwelttechnik. Was die Patentrechtler besonders beunruhigt, ist aber die Tatsache, dass die Straßburger Abgeordneten erstmals der bisher ungebremsten permanenten Ausdehnung der Rechte an „geistigem Eigentum" zentnerschwere Hindernisse in den Weg legen wollen. Das ist wirklich unerhört im Sinne der klassischen Novelle, weil einfach neu und ungewohnt. Nur so sind die heftigen Reaktionen von Patentanwälten erklärbar, die von einem „Erdbeben" im System angesichts der Signalumstellung im Parlament sprechen (Horns 2004).

Softwarepatentgegner saßen während der für die gesamte Debatte typischen Ausführungen der Gegenpartei in München wie auf Kohlen und beklagten die „Sabotage" im EU-Rat. FFII-Vorstand Hartmut klagte grundsätzlich über den Gebrauch von „Kampfbegriffen wie computerimplementierte Erfindung und geistiges Eigentum" und deckte juristische Spitzfindigkeiten in der Ratsversion auf, mit denen Programme als solche patentierbar wären. Florian Müller, Manager der Kampagne NoSoftwarePatents.com, untermauerte die Haltung der Softwarepatentgegner: „Nur eine funktionale Einheit von Gerät und Softwaresteuerung darf patentierbar sein". Noch präzisen Definitionsbedarf sieht auch Wilhelm Hoegner, EDV-Chef der besonders von der künftigen Rechtslage tangierten Stadt München, im Vorschlag des EU-Rates. Andernfalls sähen sich öffentliche Auftraggeber gezwungen, Haftungsklauseln gegenüber potenziellen Softwarepatentstreitigkeiten in Ausschreibungen einzubauen. Das hätte zur Folge, dass „nur noch große Konzerne in der Lage sein werden, den öffentlichen Markt zu bedienen. Dann haben wir wirklich eine Wettbewerbsverzerrung."

6. Ideenfang

Angesichts der geballten industriellen Marktmacht wird es dem Großteil der europäischen Softwarewirtschaft angst und bange. Im Gegensatz zu den USA ist die Landschaft hier von kleinen und mittelständischen Betrieben geprägt, die ihre Stärke zudem häufig im Bereich freier Software haben. „Aus unserer Sicht ist das Patentsystem anderer Gebiete nicht auf Software übertragbar, ohne dass es an allen Ecken und Enden zu Problemen führt", vertritt Kampagnenmanager Müller die typische Sichtweise dieses Lagers. Er führt die zu lange Laufzeit, die zu lange Bearbeitungszeit, die in Relation zum eigentlichen Entwicklungsaufwand zu hohen Kosten und die fehlende Nomenklatur, die eine Eingrenzung zu prüfender Patente sinnvoll ermöglicht, als hauptsächliche Einwände an. Da es nicht realistisch wäre, das Patentsystem an die Besonderheiten des Softwaremarktes anzupassen, „sehen wir im Urheberrecht den sinnvolleren Schutz." Das Copyright sichert unter anderem, dass Programme nicht ohne weiteres geklont werden dürfen. Es schützt die materiellen Ausführungen einer Idee, nicht diese selbst.

In der Argumentation der Softwarepatentgegner geht es bei dem Brüsseler Vorhaben just darum, Ideen rechtlich einzufangen und damit der Innovation den Garaus zu machen. Für Stallman von der *Free Software Foundation* ist die Bezeichnung Softwarepatent bereits eine Irreführung. Tatsächlich beabsichtige man, Ideen und Konzepte zu schützen, die in Programmen angewendet werden. Monopolrechtlich sanktioniert

würde also nicht eine verbesserte Waschmaschinentechnik, sondern das allgemeine Verfahren „Säubern schmutziger Kleidung". Softwarepatente seien überaus breit angelegt, sodass jedes Programm von Hunderten von Schutzansprüchen betroffen sei. Um überhaupt noch ihrer Arbeit nachgehen zu können, müssten Software-Entwickler damit künftig fast jeden Schritt von Patentanwälten prüfen lassen – oder sich auf die Gnade der Inhaber der Patentmonopole verlassen.

Ähnlich hat sich der „Verein Freie Software und Bildung" geäußert. Gemäß seiner Analyse (Heck 2004) sind die allgemeinen Auswirkungen auf die Wissensverarbeitung durch Softwarepatente bislang in der öffentlichen Diskussion untergegangen. Der erweiterte Monopolschutz für Programmcode käme dem Verein zufolge einem Verbot gleich, „ein bestimmtes Wissen im Computer zu verwenden". Ein solches Patent würde bewirken, dass das erfasste Wissen „von anderen entweder nur noch gegen Lizenzgebühren genutzt werden könnte oder auch gar nicht. Denn dem Patentinhaber steht es frei, dieses Wissen ungenutzt zu lassen: Er kann den Einsatz dieses Wissens verbieten." Da der Computer heute aber für alle ein selbstverständliches Denkwerkzeug geworden ist, würden die neuen Patentformen die Wissensverarbeitungsmöglichkeiten aller Individuen, die Freiheit des Denkens und des Forschens und somit auch von der Verfassung gewährte Grundrechte einschränken. Die politische Grundsatzentscheidung, die in Brüssel ansteht, sei somit für die Entwicklung Europas und die Freiheit seiner Individuen von fundamentaler Bedeutung.

„Der Gesetzesvorschlag zu den Softwarepatenten läuft den Interessen der zumeist mittelständischen Softwareunternehmen in Deutschland diametral entgegen", betont auch Mario Ohoven, Präsident des Bundesverbandes mittelständische Wirtschaft, mit Blick auf den EU-Rat. Er befürchtet, dass Softwareideen „zu einer Art geistiger Sperrzone erklärt werden könnten." Eine Ausweitung des Patentrechts auf Software beziehungsweise die Sanktionierung der vom Europäischen Patentamt bereits vergebenen Schutzansprüche würde auch manch größerem Unternehmen wie 1&1 zu schaffen machen. „Heute selbstverständliche Dinge wie E-Mail oder nahezu jeden anderen Dienst im Internet" sieht Achim Weiss, Entwicklungschef bei dem Netzprovider, beim Durchkommen der Richtlinie „in ihrer jetzigen Verwendung behindert". Die Branche fürchtet, dass ihr Wachstumsmotor, die rasante Weiterentwicklung der Netztechnik und der auf ihr aufbauenden Dienste auf der Basis offener Standards und durch den freien Austausch von Ideen, abgewürgt wird.

7. Innovationsförderung durch Softwarepatente?

Bezeichnenderweise führen beide Seiten in der öffentlichen Debatte das Schlagwort „Innovation" im Mund. Doch einen ernsthaften Ökonomen findet man nur schwer, der für den Patentschutz eintrat und ihn als innovationsfördernd beschrieb, ergibt sich aus der wachsenden Zahl der wissenschaftlichen Studien zum Thema. Die große Mehrzahl spricht dem Patentregime eine Innovationsförderung ab und sieht vor allem für die Volkswirtschaft keinen Gewinn daraus erwachsen. Gerade in Softwarepatenten konzentriert sich nach Auffassung des MIT-Innovationsforschers James Bessen und seines Kollegen Robert Hunt von der *Federal Reserve Bank of Philadelphia* das wach-

sende Übel, dass die staatlich gewährten Monopole wettbewerbsfeindlich eingesetzt werden (Bessen und Hunt 2004). Patente auf Computerprogramme und internetbasierte Geschäftsmethoden wie das in den USA gewährte „1-Klick-Patent" Amazon seien leicht zu erhalten, da sie ohne die Erstellung aufwändiger Prototypen auskämen. Zudem seien sie sehr weit angelegt, da sie keinen physikalischen Beschränkungen unterlägen und sich auf hunderte Passagen in Programmen beziehen könnten.

Nicht nur im Bereich Software, sondern etwa auch bei der Biotechnologie warnen Forscher verstärkt vor einer „Überhitzung" des Patentsystems. Den Markt just durch immer mehr staatliche Schutzrechte dynamisieren zu wollen, sei ein logischer Fehler. Guter Wettbewerb funktioniere über Qualität und Preis, während die Schlacht zum Aufbau immer weiterer Patentportfolios die intellektuellen Ressourcen an die falschen Prioritäten fessele. So könnte sich mancher Befürworter von Softwarepatenten in der Industrie letztlich selbst Steine in den Weg legen. Aktuelles Beispiel ist die Internet–Telefonie, die die Größten im Markt als lukratives Geschäftsfeld mit dem Potenzial zu weiteren rasanten Kostensenkungen ausgemacht haben: Der innovativen Technik gibt Professor Henning Schulzrinne von der amerikanischen Columbia University in Märkten, in denen Softwarepatente eine gesetzliche Grundlage bekommen, keine Chance. Firmen, die auf einem solchen verminten Gelände tätig werden wollten, bliebe wenig anderes übrig, als zu warten, bis die Schutzrechte ausgelaufen seien. Auch der Fraunhofer-Forscher Knut Blind bestätigt, dass eine „zusätzliche Anreizfunktion" für mehr Investitionen in die Bereiche Forschung und Entwicklung von Unternehmen durch Patente nicht nachweisbar sei (Krempl 2004*b*). Stattdessen würden Umfragen auf „steigende Rechtskosten, verunsicherte Innovatoren und Probleme für die formale Standardisierung" hinweisen.

Die negativen Nebeneffekte des Patentregimes machen sich so zwar gerade im IT–Markt langsam bemerkbar. Microsoft etwa kämpft mit einem weit gestrickten Browserpatent des Start-ups Eolas, das sich auf die Einbindung von Helferapplikationen in den Internet Explorer (und in andere Internet-Navigationswerkzeuge) bezieht, und zahlte 2004 440 Millionen US-Dollar an die inzwischen zu Sony gehörende kalifornische Firma Intertrust, um eine Streitigkeit über ein Patent rund ums Digital Rights Management aus der Welt zu schaffen. Dennoch sehen sich die IT-Giganten mit ihren zigtausend Patenten am längeren Hebel. Mit dem System arrangiert haben sich zudem internationale Player aus Europa wie Nokia oder Siemens. Sie haben dem EU-Rat mit dem Hinweis auf ihre Milliardeninvestitionen in Forschung und Entwicklung klar gemacht, dass sie weiter an der Patentierungsschraube drehen möchten.

Bislang dürfte es in Europa nach dem allgemeinen Verständnis gar keine Softwarepatente geben. Computerprogramme „als solche" sind gemäß des Europäischen Patentübereinkommens, das die Grundregeln für die Gewährung des 20 Jahre gültigen Monopolschutzes auf Erfindungen auf dem alten Kontinent festlegt, nicht patentierbar. Doch alles ist eine Frage der Definition, befindet das Europäische Patentamt in München seit 20 Jahren und hat so bereits zahlreiche Softwarepatente erteilt. Insgesamt 30 000, argwöhnen die FFII-Aktivisten, die entsprechende Beispiele in einem „Gruselkabinett" im Web sammeln. Darunter ist auch das berühmt-berüchtigte Patent auf den „Fortschrittsbalken", das dem Patentweltmeister IBM erteilt wurde. Damit ist

das gängige Verfahren geschützt, das einen Computernutzer etwa beim Installieren eines Programms über den Verlauf der Aktion aufklärt.

Aber beispielsweise auch zahlreiche Standardkomponenten von Webshops sind bereits mit zweifelhaften Patentansprüchen in Europa belegt. Etwa der elektronische Einkaufswagen: Die Rechte an dem simplen Verfahren, im Web zu erstehende Gegenstände zunächst in einer Liste zu sammeln und erst nach dem Rundgang im virtuellen Laden zu bezahlen, hat sich Sun Microsystems gesichert. Sollte ein Webhändler auf die nahe liegende, triviale Idee gekommen sein, gekaufte Waren auch als Geschenke an Dritte zu liefern, befindet er sich prinzipiell mit dem amerikanischen Netzgroßhändler Amazon im Clinch. Dem gehört seit 2003 ein europäisches Patent auf diese Methode. 18 weitere fragliche Schutzansprüche hat FFII in einem gängigen Webshop ausgemacht. Sie reichen vom Link auf ein größeres Produktfoto bis zum Verkauf von Gegenständen über ein Netzwerk an sich. „Problematisch an Softwarepatenten ist die vielfach geringe Erfindungshöhe", erläutert Joachim Henkel vom Institut für Innovationsforschung der Ludwig-Maximilian-Universität München. Durch die Vielzahl zum Teil offensichtlicher Patente entsteht außerdem eine hohe Unsicherheit. Es wird schwieriger festzustellen, ob ein Computerprogramm irgendein bestehendes (oder gar ein angemeldetes und noch nicht erteiltes) Patent verletzt. Umgekehrt schütze ein kleines Unternehmen ein eigenes Patent nur begrenzt vor Verletzungen desselben durch größere Wettbewerber, da diese im Zweifel bessere Anwälte und stärkeren finanziellen Rückhalt hätten.

Angesichts des zähen Ringens um die Richtlinie und der so gut wie nicht vorhandenen „Kompromissbereitschaft" zwischen den großen Lagern stellt sich letztlich die Frage, ob Europa ein übergreifendes Softwarepatentgesetz aus Brüssel überhaupt benötigt. Karl-Friedrich Lenz, Europarechtler an der University Aoyama Gakuin in Tokyo, warf der EU-Kommission bereits 2002 angesichts der Vorlage ihres Richtlinienvorschlags vor, einen Kropf erzeugt zu haben. Seine Argumentation: Wenn der „einheitliche Text" des Patentübereinkommens nicht verändert und an den grundsätzlichen Patentierungsregeln nichts verändert werden solle, wie von der Kommission behauptet, würde jede „Harmonisierung" in den Mitgliedsstaaten, die sich eh dem Europäischen Patentübereinkommen angeschlossen haben, nur „weitere Verwirrung stiften". Sollte das Richtlinienprojekt scheitern, würden in Deutschland weiter letztlich das Bundespatentgericht und der Bundesgerichtshof die Grenzen der Patentierbarkeit von Software festsetzen. Softwarepatentgegner hätten zwar prinzipiell eine klärende Richtlinie entlang der Vorgaben des Europaparlaments lieber, sehen diese Alternative aber auch nicht als die schlechteste an: Zumindest ist die Rechtsprechung noch nicht gefestigt und es könnten sich weitere Möglichkeiten ergeben, auf diesem Weg über die Gerichte das Problem der ungewollten Trivialpatente in den Griff zu kriegen.

Literaturverzeichnis

Bessen, J. und Hunt, R. M. (2004), 'An empirical look at software patents', *http://www.researchoninnovation.org/* . http://www.researchoninnovation.org/swpat.pdf [17.12.2004].

Blasum, H. (2004), 'Patentrecherche Linux-Basisclient München', *Foundation for a Free Information Infrastructure* . http://www.ffii.org/~blasum/basisclient/swpatmuc.pdf [17.12.2004].

Evers, M. und Traufetter, G. (2004), 'Lauf übers Mienenfeld', *Der Spiegel* **33**.

Heck, H.-J. (2004), '"Software"patente? – Eine Analyse', *Freie Software und Bildung e.V.* . http://fsub.schule.de/freie/3freie-index-swp.htm [17.12.2004].

Horns, A. H. (2004), 'BLOG@IP::JUR', *ipjur.com* . http://www.ipjur.com/2003_09_01_archive.php3 [17.12.2004].

Krempl, S. (2004*a*), 'Gefahr für den Mittelstand', *c't magazin für computertechnik* **12**, S. 60 ff.

Krempl, S. (2004*b*), 'Open Source wird sterben, wenn Softwarepatente kommen', *heise newsticker* . http://www.heise.de/newsticker/meldung/51217 [17.12.2004].

Osterloh, M., Rotha, S. und Kuster, B. (2004), Open-Source-Softwareproduktion: Ein neues Innovationsmodell?, *in* R. Gehring und B. Lutterbeck (Hrsg.), 'Open Source Jahrbuch 2004', Lehmann, Berlin, S. 277 ff.

Ravicher, D. (2004), 'OSRM POSITION PAPER: Mitigating Linux Patent Risk', *Open Source Risk Management* . http://www.osriskmanagement.com/pdf_articles/linuxpatentpaper.pdf [17.12.2004].

Sedlmaier, R. und Gegerich, J. (2004), 'Rechtliche Bedingungen und Risiken der Landeshauptstadt München für den Einsatz von Open-Source Software', *http://www.ris-muenchen.de/RII/index.jsp* . http://www.ris-muenchen.de/RII/RII/DOK/SITZUNGSVORLAGE/517379.pdf [17.12.2004].

Ude, C. (2004), 'Open Source – München fordert Klarheit', *www.muenchen.de* . http://www.muenchen.de/vip8/prod2/mde/_de/rubriken/Rathaus/40_dir/presse/2004/pressemitteilungen/linux_pressepapier.pdf [17.12.2004].

Wiebe, A. (2004), Patentschutz und Softwareentwicklung - ein unüberbrückbarer Gegensatz?, *in* R. Gehring und B. Lutterbeck (Hrsg.), 'Open Source Jahrbuch 2004', Lehmann, Berlin, S. 121 ff.

Tragen die Juristen Open-Source-Software zu Grabe? – Die GNU GPL vor Gericht

THOMAS EBINGER

Das Recht bedroht den zum Riesen heranwachsenden Zwerg Open Source. Es bestehen im Wesentlichen zwei Gefahren: (a) die fehlende Durchsetzbarkeit von Open-Source-Lizenzen in Deutschland und (b) die Verletzung von softwarebezogenen Patenten durch Open-Source-Software (OSS). Beide Bedrohungen könnten das etablierte Softwareentwicklungsmodell in den Grundfesten erschüttern und das Ende von OSS einläuten. Die GNU General Public License (GPL) ist die bekannteste Open-Source-Lizenz. Deren Besonderheiten – sie ist nach US-amerikanischem Recht erstellt, liegt offiziell nur in englischer Sprache vor und die hierunter entwickelte Software wird international im Internet entwickelt – bergen gewisse rechtliche Schwierigkeiten bei ihrer Anwendung in Deutschland. Mit dem Urteil des Landgerichts München I vom 19.05.2004 liegt nun eine erste Entscheidung eines deutschen Gerichts zur GPL vor. Die Richter befassen sich u. a. mit folgenden Fragen: Kann die Nutzung einer GPL-Software untersagt werden, sofern gegen bestimmte Klauseln der GPL verstoßen wird? Wer kann Rechte aus OSS (gerichtlich) geltend machen? Unter welchen Voraussetzungen gilt die GPL? Gilt die Lizenz in der deutschen Rechtsordnung trotz des englischen Lizenztextes? Das Mehr an Rechtssicherheit durch diese Gerichtsentscheidung wird kontrastiert durch die ungelöste Bedrohung von OSS durch Softwarepatente.

1. Einleitung

Was andere nicht geschafft haben, besorgen nun die Juristen? Juristische Veröffentlichungen haben eine Kontroverse über erhebliche Risiken beim Einsatz von Open–Source-Software (OSS) entfacht (vgl. heise newsticker 2003*b*, Linux Verband 2003, Ebinger 2003). So schreibt der Verband der Softwareindustrie Deutschlands e. V., gestützt auf ein professorales Rechtsgutachten (Spindler 2003), an einige Ministerien und Behörden:

> „Der VSI sieht . . . bestätigt, dass aus rechtswissenschaftlicher Perspektive Risiken beim Einsatz von OSS existieren. Diese haben eine

> unmittelbare unternehmenskritische Relevanz, da erhebliche wirtschaftliche Nachteile durch die mangelhafte Durchsetzung urheber-, vertrags- und haftungsrechtlicher Ansprüche entstehen können." (heise newsticker 2003*a*)

Tragen daher die Juristen Open-Source-Software zu Grabe? Ist die GPL nicht den Speicherplatz wert, den sie belegt? Wird OSS wie ein Kartenhäuschen in sich zusammenfallen, weil die GPL rechtlich belanglos ist und wird daher keine weitere OSS entwickelt werden wird?

Vielfach haben sich zwischenzeitlich juristische Veröffentlichungen mit OSS beschäftigt. Derartige Publikationen von Juristen sind wichtig. Sie verstauben aber schnell in den Bibliotheken als dogmatische Fingerübungen, falls sich die Rechtsprechung diesen Auffassungen nicht anschließt. Daher wurde mit großer Aufmerksamkeit die Entscheidung des Landgerichts München I vom 19. Mai 2004 aufgenommen, als eine der ersten Gerichtsentscheidungen über mehrere Fragen zu der prominentesten OSS–Lizenz, der *GNU General Public License* (GPL): Können bestimmte Klauseln der GPL gegenüber Verwendern von GPL-Software durchgesetzt werden? Diese und weitere Fragen hatten die Richter zu klären. Deren Entscheidung schaffte eine erste Klärung in einigen, unter Juristen umstrittenen Punkten.

2. Wie kam es zu dem Gerichtsurteil?

Vor dem Landgericht München I standen sich als Antragsteller ein Leiter eines OSS–Entwicklerteams (*head of core team*) und als Gegnerin ein Unternehmen gegenüber. Das Unternehmen bot neben anderen Netzwerkprodukten einen drahtlosen Router (Wireless Router) an. Dieser Router nutzte die Software *netfilter/iptables*[1] des OSS–Entwicklerteams, die besonders für Firewalls von großer Bedeutung ist. Wie bei dem überwiegenden Anteil der OSS wird diese Software klassisch im Internet von mehreren Entwicklern erstellt und weiterentwickelt. Diese netfilter/iptables Software haben die Entwickler unter der GNU General Public License (GPL)[2] freigegeben bzw. lizenziert. Die GPL ist eine der bekanntesten Open-Source-Lizenzen, da insbesondere GNU/Linux unter dieser Lizenz verbreitet wird. Sie ist aber auch eine der umstrittensten.[3]

Sie schreibt insbesondere vor, dass die Hinweise auf die Lizenzierung unter der GPL beizubehalten sind, die GPL-Lizenz beizufügen ist (Ziffer 1 Abs. 1 GPL) und mit der Software regelmäßig auch der Source-Code angeboten werden muss (Ziffer 3 a) GPL).

> „*1.* You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you

1 Homepage: http://www.netfilter.org/.
2 Vgl. http://www.gnu.org/copyleft/gpl.html [18. Nov 2004].
3 Vgl. heise newsticker (2001).

> conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
>
> *3.* You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
>
> *a)* Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange;. . . "

Der Router des Unternehmens nutzte die Software des OSS-Projekts im ausführbaren Binärcode (Objektcode) ohne Hinweis auf die GPL. Mit dem Router erhielt der Käufer die ausführbare Software ohne den Quellcode (source code). Im Internet ermöglichte das Unternehmen den Download der ausführbaren Software (Objektcode), nicht aber den Download des Quellcodes, entgegen der Verpflichtung aus der GPL. Auch hatte das Unternehmen weder darauf hingewiesen, dass die Software der GPL unterliegt, noch den GPL Lizenztext zugänglich gemacht.

Das Unternehmen war nicht der einzige Übeltäter, der GPL-Software unter Verstoß gegen die GPL nutzte und vertrieb. Doch die anderen Unternehmen lenkten nach Hinweisen auf den Verstoß gegen die GPL außergerichtlich ein. Nicht so das Unternehmen, dem die Ehre des richtungsweisenden Gerichtsurteils zukommt. Nachdem eine außergerichtliche Klärung des Rechtsstreits scheiterte, strengte der in Deutschland wohnende Leiter des Softwareprojekts *netfilter/iptables* das vorliegende Gerichtsverfahren vor dem Landgericht München I an, um dem Unternehmen die Nutzung und Verwertung der Software in der bisherigen Weise zu untersagen.

3. Die GPL und das deutsche Recht

Ende der 1980er Jahre in den USA geschaffen, sollte die GNU General Public License Software dauerhaft frei verfügbar halten. Die spätere internationale Bedeutung der GPL konnte man damals noch nicht vorhersehen und hatte daher die Untiefen deutschen Rechts nicht berücksichtigt. In deutschen Veröffentlichungen bis Mitte der 90er Jahre setzte man sich allenfalls am Rande mit Public-Domain-Software und Freeware auseinander (vgl. Müller-Broich 1998). Eine rechtliche Auseinandersetzung mit der GPL setzte aber erst Ende der 90er Jahre ein,[4] als die Bedeutung dieser Lizenz offenkundig wurde und der Begriff „Open Source" mit offizieller Definition geprägt wurde (vgl. Open Source Initiative 2004*a*,*b*). In den letzten Jahren nahmen die

4 Vgl. Gehring (1996), die Studienarbeit eines Informatikers, Metzger und Jaeger (1999), Siepmann (1999), Sester (2000), Koch (2000*a*,*b*).

rechtlichen Publikationen zu Open-Source-Software mit deren Bedeutungszuwachs zu.[5] Heutzutage interessieren die Untiefen des deutschen Rechts sehr wohl. Von interessierter Seite schürte man rechtliche Bedenken gegen die Vereinbarkeit der GPL mit dem deutschen Recht,[6] erhielt aber auch Widerspruch.[7] Diese Meinungen, ob pro oder contra, werden aber bedeutungslos, wenn die Rechtspraxis eine andere Sprache spricht. Daher liegt nun mit Urteil des Landgerichts München ein wichtiger, erster Meilenstein für die Anerkennung und Durchsetzung der GPL vor.

Die GPL – Als Tiger gesprungen, im Gericht als Bettvorleger gelandet?

Für seine Entscheidung über die GPL hatte das Gericht sich mit verschiedenen Fragekomplexen zu beschäftigen, die für OSS auch in anderen Fallgestaltungen von großer Bedeutung sind.

3.1. Kann sich GPL-Software auf das Urheberrecht berufen?

Proprietäre Software legt den Nutzern üblicherweise große Beschränkungen durch das Urheberrecht (englisch: Copyright) auf. Die GPL-Lizenz dagegen räumt den Nutzern unter Verzicht auf die üblichen urheberrechtlichen Beschränkungen weitreichende Freiheit in der Verwendung der Software ein und nennt das *Copyleft* (vgl. Stallman 1996, 1999). Aufgrund dieses Verzichts wurde daher vereinzelt die Frage gestellt, ob mit der GPL ein Verzicht auf den Schutz durch das Urheberrecht erfolgt sei. Das wies das Gericht zurück und bestätigt damit auch die Auffassung des Schöpfers der GNU General Public License, der *Free Software Foundation* (FSF). Danach wird das Urheberrecht genutzt, um die freie Verfügbarkeit der Software abzusichern (vgl. Landgericht München 2004, S. 12). Einem Entwickler von GPL-Software kann daher nicht der urheberrechtliche Rechtsschutz verwehrt werden.

Als Zwischenergebnis bedeutet dies, niemand darf die Software des Softwareprojekts nutzen, es sei denn, die Entwickler haben dies dem Nutzer gestattet. Dafür muss das hier betroffene Unternehmen Nutzungs- und Verwertungsrechte an der Software von den Urhebern erworben haben, die es dem Unternehmen erlauben, die Software zu nutzen. Das Gericht bezeichnet das urheberrechtliche Nutzungs- und Verwertungsrecht auch als *dingliches* Recht in Abgrenzung zu einem Recht aus einem Vertrag, das regelmäßig nur gegenüber dem Vertragspartner gilt. Das versteht man nur, wenn man etwas mit den Besonderheiten des deutschen Softwarerechts vertraut ist. Es unterscheidet Ebenen beim Erwerb von Software. Der Nutzer hat ein *dingliches* (eigentumsähnliches) Recht zur Nutzung der Software von dem Urheber und einen Vertrag mit seinem unmittelbaren Vertragspartner. Wer also einen PC mit Microsoft Windows im Handel kauft, hat einen Vertrag mit seinem PC-Händler und erhält daraus

5 Vgl. Ebinger (2002) und Spindler (2004) mit weiteren Nachweisen.

6 Vgl. heise newsticker (2003*a*) und Spindler (2004), dort vor allem S. 104–106 (Zusammenfassung: urheber-, vertrags-, haftungsrechtliche Probleme).

7 Vgl. Linux Verband (2003), Free Software Foundation Europe (2003) sowie Ebinger (2003) mit einer Kurzkritik.

von Microsoft – als Urheber der Windows-Software – ein dingliches Nutzungsrecht an der Software.

3.2. Wer kann gegen den Verletzter der GPL klagen?

In den juristischen Veröffentlichungen wird umfangreich die Frage problematisiert, wer Urheberrechte aus der GPL-Software geltend machen kann (vgl. Spindler 2004, S. 26–38 mit weiteren Nachweisen). Erstellt ein Entwickler eine Software, dann ist er der Urheber und erwirbt hieran Urheberrechte (§§ 7, 2 Abs. 1 Ziffer 1, 69a ff. Urhebergesetz (UrhG)). Erstellen mehrere Entwickler eine Software in nicht voneinander trennbaren Teilen, sind alle Entwickler Urheber, oder wie das Gesetz in § 8 UrhG sagt, Miturheber. Wird in einem Open-Source-Projekt wie üblich gemeinsam nach dem „Bazaar"-Modell (Raymond 1998) entwickelt, müssen dann alle weltweit verteilten Entwickler ein Gerichtsverfahren gemeinsam einleiten? Das könnte oftmals ein Fall der Unmöglichkeit sein.

In dem vorliegenden Fall hat der in Deutschland wohnende Leiter des Softwareprojekts als Einzelperson das Gericht angerufen. Das Gericht hat diese Frage nicht sonderlich problematisiert und ihm ohne großen Kommentar einen Verbotsanspruch nach § 8 Abs. 2 UrhG (zumindest) als Miturheber gegen das Unternehmen zugestanden. Nach § 8 Abs. 2 S. 3 UrhG kann jeder Miturheber die Verletzung des Urheberrechts aller geltend machen. Will der Kläger aber auch z. B. Schadensersatz wegen der Verletzung, muss er aber die Zahlung an alle fordern. Das heißt, will man lediglich die weitere Verletzung der GPL untersagen, ist es nicht erforderlich, dass alle beteiligten Entwickler klagen. Es muss auch nicht unbedingt der Softwareprojektleiter sein, solange nur nachweisbar ist, dass der Kläger (oder Antragsteller) einen Teil der aktuellen Software selbst entwickelt hat.

Das Gericht hat indirekt deutlich gemacht, dass es den antragstellenden Entwickler nicht nur als Miturheber, sondern auch als Einzelurheber ansehen könnte. Denn es schreibt, er sei „(zumindest) Miturheber". Der Maintainer hatte einen Teil der Software selbst geschrieben. Sofern diese Teile von der anderen Software abtrennbar sind, hätte er gegebenenfalls sogar als alleiniger Urheber Schadensersatzansprüche gegen das rechtsverletzende Unternehmen geltend machen können. Dabei verliert ein Entwickler seine weitergehenden Rechte als Alleinurheber nicht, wenn weitere Entwickler die Software nur unbedeutend überarbeitet haben.

3.3. Findet deutsches Recht Anwendung?

Nach welcher Rechtsordnung ist der Fall zu entscheiden? Muss wegen der US–amerikanischen Lizenz das Recht des zuständigen US-amerikanischen Bundesstaates angewandt werden? Ist nach dem Recht des Wohnsitzes des deutschen Maintainers, dem Sitz des Unternehmens oder des (fernöstlichen) Herstellers des Routers zu entscheiden?

Das Gericht verwies auf eine Entscheidung des Bundesgerichtshofes.[8] Danach ist

8 BGH, Urteil vom 2.10.1997 – I ZR 88–95 (München); NJW 1998, S. 1395–1398 (*Spielbankaffaire*).

das Recht des Staates anzuwenden, für den die urheberrechtlichen Ansprüche geltend gemacht werden (urheberrechtliches Schutzlandprinzip). Da in dem vorliegenden Fall die Verbotsrechte aus der GPL für Deutschland geltend gemacht werden, sahen die Richter keine Bedenken in der Anwendung des deutschen Rechts.

3.4. Sind die Klauseln der GPL Allgemeine Geschäftsbedingungen?

Das Gericht prüft, ob die Regelungen der GPL, aus denen der Entwickler Rechte herleitet, mit den Sonderregelungen für Allgemeine Geschäftsbedingungen (AGB) vereinbar sind. Allgemeine Geschäftsbedingungen liegen häufiger vor als man denkt. Insbesondere muss über Regelungen nicht „AGB" oder „Allgemeine Geschäftsbedingungen" stehen. Darunter fallen auch Formularverträge, die für eine Vielzahl von Vereinbarungen bestimmt sind (§ 305 BGB). Die GPL-Lizenz wird nicht mit jedem Nutzer einzeln ausgehandelt, sondern nur nach dem Prinzip „take it or leave it" angeboten. Daher ist die GPL als ein Formularvertrag anzusehen und muss den rechtlichen Anforderungen an Allgemeine Geschäftsbedingungen (§§ 305 ff. BGB) standhalten. Die Überprüfung der GPL als Allgemeine Geschäftsbedingung stellt vor allem die Wirksamkeit des weitgehenden Haftungsausschluss der GPL in Frage.

Gelten die Klauseln der GPL oder hat das Unternehmen etwa ein Nutzungsrecht an der Software ohne die Beschränkungen der GPL erlangt?

Die Richter fragen nun weiter, ob die GPL als Allgemeine Geschäftsbedingungen wirksam in das Rechtsverhältnis zwischen dem Unternehmen und den Entwicklern einbezogen wurde oder ob die Regelungen der GPL zwischen den Parteien gar nie wirksam wurden. Damit die GPL als AGB überhaupt gelten können, musste dem Unternehmen die Kenntnisnahme der GPL möglich und zumutbar gewesen sein (§ 305 Abs. 2 BGB). Was heißt das?

Muss eine offizielle GPL-Lizenz in Deutsch zur Verfügung gestellt werden?

Die Richter problematisierten zu Recht, ob die Geltung der GPL zwischen den Parteien daran scheitert, dass die GPL-Klauseln nur in englischer Sprache angeboten werden, es aber keine offizielle deutsche Version gibt. Viele Deutsche können geltend machen, dass sie Englisch nicht oder nicht gut verstehen und schon gar nicht das juristische Englisch der Lizenz. Das deutsche Recht sieht für die Geltung von AGB regelmäßig vor, dass diese zu ihrer Wirksamkeit in deutscher Sprache angeboten werden müssen. Dann wäre das Ergebnis: Das Unternehmen ist nicht an die GPL gebunden. Nicht so aber das Gericht. Es bejaht die Einbeziehung der GPL in das Rechtsverhältnis der Parteien trotz englischsprachiger Lizenz ohne offizielle deutsche Übersetzung der GPL. Zur Bejahung der Einbeziehung der GPL stellt das Gericht darauf ab, dass auf der Website des Softwareprojekts auf die GPL hingewiesen wurde und diese dort in englischer Sprache aufgerufen werden kann. In dem konkreten Fall verweisen die Richter darauf, dass Englisch in der Computerindustrie gängige

Fachsprache sei und deshalb das Unternehmen als gewerbliches Softwareunternehmen die englischen GPL gegen sich gelten lassen müsse.

Ob die GPL auch wirksam in das Rechtsverhältnis einbezogen worden wäre, wenn der Nutzer die Software nur zu privaten Zwecken in Deutschland heruntergeladen hätte, können die Richter in diesem Fall offen lassen. Diese Frage ist alles andere als unproblematisch. Für die Entwickler ist aber von praktischer Bedeutung hauptsächlich der Missbrauch und Verstoß gegen die GPL durch kommerzielle Nutzer.

Was, wenn ein deutscher Lizenztext erforderlich wäre?

Das Gericht hatte daher im Ergebnis keine Zweifel, dass das Unternehmen die GPL trotz englischer Sprache in zumutbarer Weise zu Kenntnis nehmen konnte und sie daher gilt (vgl. Landgericht München 2004, S. 14).

Ob die Kenntnisnahme zumutbar möglich war, kann aber aus weiteren Gründen in Frage gestellt werden (siehe unten, Abs. 6.1., S. 262). Doch selbst für diesen Fall würden die Richter – so ihre Hilfsüberlegung – dem Softwareentwickler Recht geben wollen. Dabei beziehen sie sich auf eine Besonderheit des Rechts zu den Allgemeinen Geschäftsbedingungen (§§ 305 ff. BGB). Werden die besonderen Anforderungen an die Einbeziehung solcher Formularverträge nicht beachtet, so gelten diese Klauseln nicht. Das führt aber im Regelfall nicht dazu, dass das Geschäft „platzt". Vielmehr bleibt das Rechtsverhältnis wirksam. Der Vertrag ist im Normalfall nicht gescheitert im Sinne von „Du akzeptierst meine AGB oder es gibt keine Vereinbarung" (Friss–oder-stirb-Prinzip), sondern es gelten dann die gesetzlichen Auffangregeln. Das ist für den Verwender von AGB im Regelfall von Nachteil, da der Verwender sich mit seinen AGB einen Vorteil gegenüber der Gesetzeslage verschaffen will. Das mag hier anders liegen, da die Entwickler nicht so sehr sich, sondern den Nutzern einen Vorteil verschaffen wollen. Deshalb deutet das Gericht an, dass aufgrund der Besonderheiten der GPL selbst beim Scheitern der Einbeziehung der GPL in das Rechtsverhältnis der Parteien, ausnahmsweise das Unternehmen keinerlei Rechte erworben hätte. Wenn also die GPL nicht gilt, dann soll – so das Gericht – der Softwareerwerber auch kein Nutzungsrecht haben.

3.5. Anforderungen an die Zurverfügungstellung von GPL-Software

Die GPL besagt, wer anderen GPL-Software zur Verfügung stellt, muss nach der GPL unter anderem den Verweis auf die Geltung der GPL beibehalten (Ziffer 1 Abs. 1 GPL), eine Kopie des GPL-Lizenztextes der Software beifügen (Ziffer 1 Abs. 1 GPL) und neben dem ausführbaren Code (Objektcode) auch die Software im Source Code (Quellcode) beifügen (Ziffer 3 GPL).[9] Welche Auswirkung hat nun ein Verstoß gegen diese Verpflichtungen?

9 Der vollständige Text der GPL ist im Anhang abgedruckt.

3.6. Rechtsverlust – Eine unangemessene Benachteiligung des Nutzers?

Wer diese Bedingungen der GPL nicht einhält, darf die Software nicht nutzen. Das sieht Ziffer 4 GPL vor:

> „4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance."

Der Verlust jeglicher Rechte an der Software ist ein großer Einschnitt. Die Verwertung der Software kann dann sogar eine Straftat darstellen (§ 106 UrhG). Zumindest bei proprietärer Software werden regelmäßig die Verletzungen von Lizenzbestimmungen nicht zum Verlust des urheberrechtlichen Nutzungsrechts an der Software führen. Rechtsfolge ist meist lediglich eine Vertragsverletzung, die nur Ansprüche zwischen den Vertragspartnern, nicht aber Ansprüche des Softwareherstellers/-entwicklers gegenüber den Dritt- und Vierterwerbern der Software begründen würde, mit denen er keine Vertragsbeziehung hat.[10]

Danach hätte das Unternehmen durch die Verstöße gegen die GPL nicht das Recht zur Nutzung der Software verloren. Von der Regel – Lizenzverletzungen führen nicht zum Verlust der Nutzungsrechte – kennt der Jurist natürlich, wie sollte es anders sein, Ausnahmen.

Lizenzierung einer Software unter der GPL als wirtschaftlich eigenständige Nutzungsart?

Der Urheber kann Nutzungsrechte nur für eine bestimmte Nutzungsart einräumen, so dass jedem Erwerber die Nutzung des Werkes in einer anderen Nutzungsart verboten bleibt. Die Online-Veröffentlichung eines Artikels oder eines Fotos ist gegenüber der Print-Veröffentlichung eine neue, andere Nutzungsart. Der Autor, der z. B. einer Zeitung nur Print-Rechte eingeräumt hat, kann die Online-Veröffentlichung untersagen. Übersetzt auf die GPL bedeutet dies: Sieht man in einer unter der GPL lizenzierten Software eine eigenständige Nutzungsart, so ist ein Verstoß gegen die Bedingungen der GPL eine urheberrechtlich unerlaubte Nutzung in einer nicht eingeräumten Nutzungsart. Der Vorteil dabei: Eine solche Nutzung kann der Urheber gegenüber jedem untersagen, auch wenn keine Vertragsbeziehung mit dem GPL-Verletzer besteht. Das Gericht überprüfte also, ob die unter der GPL lizenzierte Software eine wirtschaftlich eigenständige Nutzungsform ist.

Hierbei verwiesen die Richter auf die OEM-Entscheidung des Bundesgerichtshofs (BGH).[11] Das Microsoft-Betriebssystem Windows konnte man entweder isoliert als

10 BGH, Urteil vom 6.7.2000 – I ZR 244/97 (KG), NJW 2000, S. 3571–3573 (*OEM-Lizenz von Microsoft*).
11 BGH, Urteil vom 6.7.2000 – I ZR 244/97 (KG), NJW 2000, S. 3571–3573 (*OEM-Lizenz von Microsoft*).

reine Software erwerben oder wie häufig verbunden mit einem PC. Die mit einem PC vertriebene Software, als sogenannte OEM-Software bezeichnet, kostete deutlich weniger als die isoliert ohne PC vertriebene Windows Software. Es lohnte sich also, OEM-Software von dem PC zu trennen und diese isoliert weiterzuverkaufen. Das wollte Microsoft dem Händler – mit dem Microsoft keine vertraglichen Beziehungen hatte – gestützt auf die OEM-Lizenz – untersagen. Microsoft hatte argumentiert, wer sich nicht an die Bedingungen der OEM-Lizenz hält, hat daran das urheberrechtliche Nutzungsrecht verloren. Denn eine OEM-Software sei eine eigenständige Nutzungsart. Der BGH verneinte eine Verletzung der Urheberrechte von Microsoft, weil durch den Verkauf der Software Microsoft grundsätzlich keine Urheberrechte mehr geltend machen konnte. Denn die Urheberrechte von Microsoft an der Software auf dem Datenträger hätten sich mit dessen regulären Verkauf erschöpft (Erschöpfungsgrundsatz). Vertragsrechtliche Ansprüche griffen in dem OEM-Fall nicht, da der verklagte Händler nur einen Vertrag mit einem Zwischenhändler, aber nicht mit Microsoft hatte. Das heißt im Ergebnis zumindest bei proprietärer Software: Der Nutzer erwirbt gegen Zahlung eines Kaufpreises eine Software auf einem Datenträger, über die er wie ein Eigentümer verfügen können soll.[12] Insbesondere darf der Softwareentwickler/das Softwareunternehmen nicht dem Nutzer vorschreiben können, ob er die Software wieder an einen Dritten und in welcher Form verkaufen kann. Zur Bejahung einer eigenständigen Nutzungsart fordert der BGH in der OEM-Entscheidung eine übliche, technisch und wirtschaftlich eigenständige und damit klar abgrenzbare Nutzungsform.

Auf die OEM-Entscheidung gestützt, lehnte das Münchener Gericht es ab, in der unter der GPL lizenzierten Software eine eigenständige Nutzungsart zu sehen und verwies dabei auf Ziffer 2 GPL (Landgericht München 2004, S. 14 f.). Hierbei hätte man mit etwas Begründungsaufwand aber auch zu einem anderen Ergebnis kommen können.[13] Letztlich kam es in der vorliegenden Entscheidung aber nicht darauf an. Denn auch ohne Bejahung einer eigenständigen Nutzungsart kam das Gericht zu demselben Ergebnis, dass die Lizenzverletzung ausnahmsweise zum Verlust der Nutzungsrechte des Unternehmens führt. Denn das Gericht bejahte eine weitere Ausnahme, eine „auflösend bedingte dingliche Einigung".

„Nutzungsberechtigung nur unter der Bedingung, dass. . . ?"

Was ist eine „auflösend bedingte dingliche Einigung"? Eine Verletzung der GPL ist danach nicht nur eine Vertragsverletzung, sondern führt zum Verlust des urheberrechtlichen Nutzungsrechts an der Software. Unabhängig von der vertraglichen Rechtsbeziehung erwirbt der Nutzer das Recht zur Nutzung der Software unter der Bedingung (Ziffer 4 S. 2 GPL), dass er die streitigen Klauseln der GPL einhält. Tut er das nicht, führt das nicht nur zu einer Vertragsverletzung, sondern auch zum Verlust des Nutzungsrechts an der Software. Diese Argumentation erlaubt den Software-

12 Über Datenträger mit nur abgespeckten Softwareversionen (Recovery-CD-ROM) war nicht zu entscheiden.

13 Auch, wenn weite Teile der juristischen Literatur dies ablehnen.

entwicklern, ihre Rechte gegenüber jederman geltend zu machen, ohne dass sie ein Vertragsverhältnis mit dieser Person nachweisen müssten.

Bedingte Einigung – eine unangemessene Benachteiligung des Softwareerwerbers?

Damit ist die Angelegenheit aber noch nicht erledigt. Denn sonst könnte durch eine solche Bedingung der Schutz des Nutzers leicht ausgehebelt werden. Gerade die OEM-Entscheidung des BGH will verhindern, dass der Softwareverkäufer den Softwareerwerber einseitig ans Gängelband nimmt und ihm besonders vorschreibt, an wen und wie der Erwerber die Software weitergeben darf. Daher überprüft das Münchener Gericht auch, ob nicht durch eine solche bedingte Einräumung des Nutzungsrechts die Rechte der Nutzer der Software – hier die Rechte des Unternehmens – umgangen werden.

Bei der Prüfung, ob eine Umgehung des Schutzes des Nutzers vorliegt, stellt das Gericht vorrangig auf den Zweck der Regelung im Urheberrecht ab. Durch die Einräumung eines Nutzungsrechts soll bei einer mehrstufigen Vertriebskette - Softwareentwickler, Großhändler, Zwischenhändler, Ersterwerber, Zweiterwerber - keine Rechtsunsicherheit eintreten. Es besteht die Gefahr, dass ein Veräußerer (Verkäufer oder auch Schenker) einem Erwerber eine Software zur Verfügung stellt, woran der Veräußerer (zwischenzeitlich) keine Rechte mehr hat. Die Folge ist, dass der Erwerber auch keine Rechte erwerben kann. Die Richter warnen hier vor einer Einschränkung der *Verkehrsfähigkeit* der Software. Dieses Prinzip besagt, wer eine Software erwirbt, soll darauf vertrauen können, dass er die notwendigen Rechte hierzu auch bekommt. Das Nutzungsrecht des Softwareerwerbers soll danach nicht verloren gehen, weil einzelne Klauseln einer Lizenz – vor allem von dem Verkäufer – nicht eingehalten werden. Die Rechtsübertragungskette soll nicht abreißen und der letzte soll nicht der Dumme sein, weil einer der Rechtsvorgänger rechtswidrig handelt. Denn diesen Verstoß kann der (Letzt-) Erwerber der Software regelmäßig nicht erkennen. Hier sieht das Gericht die nachteiligen Folgen des Verstoßes gegen die GPL vorrangig bei dem Verletzer der GPL, der weniger oder nicht schutzwürdig ist. Ein Nutzer der GPL-Software ist nicht von einem Händler abhängig, er kann sich die Nutzungsrechte unmittelbar von den Entwicklern einräumen lassen. Deshalb kommt das Gericht zu dem Ergebnis, dass die Folgen einer Verletzung der GPL in erster Linie den Verletzer und nicht ahnungslose Dritte treffen. Das Gericht sah daher die Verkehrsfähigkeit durch die GPL als nicht gefährdet an. Denn Beschränkungen der Verkehrsfähigkeit der GPL-Software durch die bedingt eingeräumten Nutzungsrechte können wieder weitgehend ausgeglichen werden. Die Richter stellen dabei stark auf die Besonderheiten der GPL ab. Denn sobald der Nutzer sich wieder an die Bedingungen hält, kann er von den Entwicklern oder Dritten das Nutzungsrecht an der Software erneut erwerben. In anderen Worten: Wer von dem Unternehmen die Software ohne Quellcode und GPL-Lizenztext heruntergeladen hat, mag zunächst kein Nutzungsrecht erworben haben. Lädt sich der Nutzer bei dem Software-Projekt die GPL und den Quellcode herunter und stellt er dann die Software zum Download auf seine eigene Website mit Hinweis auf die

GPL, hat er spätestens dann wieder ein Nutzungsrecht an der Software.[14]

Im Ergebnis sieht das Gericht demnach keine Umgehung der Rechte des Nutzers und anderer Betroffener (Landgericht München 2004, S. 15–18). Daher kann es ausnahmsweise den Wegfall der Nutzungsrechte an der Software bejahen, sofern gegen die obigen GPL-Klauseln verstoßen wird.

3.7. Weitergabe mit Source-Code – Ziffer 2, 3 GPL

Die Richter sehen auch keine Bedenken gegenüber der Verpflichtung, dass die Weitergabe der Software an Dritte „kostenfrei" mit Quellcode erfolgen und dies auch für Änderungen an der Software geschehen muss (Ziffer 2, 3 GPL). Dabei stellten sie darauf ab, dass der Nutzer die Software auch „kostenfrei" erhalten hat und der Gesetzgeber das Open-Source-Prinzip jüngst durch eine Gesetzesänderung in § 32 Abs. 3 S. 3 UrhG anerkannt hat. Die gesetzliche Regelung bezieht sich zwar nicht auf die entscheidungserhebliche Frage (vgl. Hoeren 2004). Der Gesetzgeber hat aber durch diese Regelung zum Ausdruck gebracht, dass das Urheberrecht angemessen Rücksicht auf die Besonderheiten von OSS zu nehmen hat. Genau dies hat das Gericht mit dem Verweis auf den Rechtsgedanken in § 32 Abs. 3 S. 3 UrhG getan.

3.8. Hilfsüberlegung – Nichtigkeit der gesamten Vereinbarung?

Die Argumentation des Gerichts ist keineswegs unanfechtbar. Denn die Dogmatik des deutschen Urheberrechts und des Rechts zu Allgemeinen Geschäftsbedingungen berücksichtigen nicht ausreichend die Besonderheiten der GPL und den nichtkommerziellen Ansatz des GPL-Modells (siehe unten, Abs. 5., S. 260). Aber die Münchener Richter haben auch für den Fall der Ablehnung ihrer Argumentation durch ein höheres Gericht Vorsorge getragen. Sie haben ihre Entscheidung durch eine Hilfsargumentation abgesichert. Hierzu bejaht das Gericht hypothetisch eine unangemessene Benachteiligung der Softwareerwerber durch die streitigen GPL-Klauseln. Das hätte die Unwirksamkeit dieser GPL-Klauseln als unzulässige AGB-Bestimmungen zur Folge (§ 307 BGB). Dagegen würden im Regelfall alle anderen Regelungen, die sich nicht als unwirksam erweisen, wirksam bleiben (§ 306 Abs. 1 BGB). Das heißt, der Nutzer würde sein Recht zur Nutzung der Software regelmäßig nicht verlieren.[15] Er könnte die Software weiternutzen und könnte grundsätzlich weiterhin gegen die GPL verstoßen. Hier findet das Gericht aber, dass eine Ausnahme greifen müsse. Die Ziffern 3, 4 GPL seien von so prinzipieller Bedeutung für die offene Weiterentwicklung der GPL-Software, dass ausnahmsweise das gesamte Rechtsverhältnis zwischen den Entwicklern und dem Unternehmen unwirksam sei (§ 306 Abs. 3 BGB).[16]

14 Hier kann auch auf Ziffer 4 S. 3 GPL verwiesen werden.

15 Ob der Softwarenutzer dann aber die weitgehenden Rechte der GPL zur Vervielfältigung, Verbreitung und Änderung der Software noch hätte, steht auf einem anderen Blatt.

16 Vgl. oben, Abs. 3.4., S. 254.

4. Ergebnis

Damit kommt das Gericht zu dem Ergebnis: Durch die Verstöße gegen die GPL verliert das Softwareunternehmen das Recht zur Nutzung der GPL-Software. Der Softwareprojektleiter konnte dem Unternehmen die Softwarenutzung untersagen, soweit es weiterhin gegen die GPL verstößt. Besonders bei Vertrieb der Software ist auf die Lizenzierung unter der GPL hinzuweisen und die Software mit Source Code und GPL-Lizenztext lizenzgebührenfrei beizufügen. Die GPL kann also grundsätzlich auch unter deutschem Recht Geltung beanspruchen und sie erlaubt auch die Durchsetzung – zumindest gewisser – Klauseln gegenüber (kommerziellen) Nutzern der GPL-Software. Nach dem Gerichtsurteil müssen sich GPL-Entwickler die missbräuchliche Verwendung ihrer Software durch Unternehmen nicht mehr gefallen lassen und haben ein wirksames Mittel, um ihre Rechte gerichtlich, aber auch außergerichtlich durchzusetzen. Unternehmen, die ihre Ernte ohne eigene Aussaat gerne auf den fremden Feldern von GPL-Softwareprojekten einholen, gehen ein hohes rechtliches und finanzielles Risiko ein.

5. Bewertung

Nun, die Entscheidung kann man ausschließlich nach der bisherigen Rechtsdogmatik bewerten. Das greift aber zu kurz.

5.1. Dogmatische Kritik – Die reine Lehre

Aus Sicht der aktuellen Rechtsdogmatik kann man die Gerichtsentscheidung in mehreren Punkten kritisieren: so z. B. die umfassende Anwendung deutschen Rechts[17] oder die unzureichende Berücksichtigung des Erschöpfungsgrundsatzes.[18] Es ist nicht zu bestreiten, dass die Anwendung der aktuellen Rechtsdogmatik zum Urheberrecht und zu den Beschränkungen von Allgemeinen Geschäftsbedingungen nicht ausreichend die Besonderheiten von OSS berücksichtigen kann. Manche Autoren haben daher eine erhebliche Rechtsunsicherheit festgestellt.[19] Das ist aber eine rückwärts gewandte Betrachtungsweise nach dem Prinzip: „Kenn' wir nich'! Passt nicht in unser System!"

17 Die alleinige Heranziehung deutschen Rechts für die vorliegende Entscheidung ist nicht unproblematisch. Bei der urheberrechtlichen Bewertung durften die Richter zwar deutsches Recht problemlos anwenden. Jedoch prüft das Gericht im Folgenden verschiedene vertragsrechtliche Fragen, die nach geltendem deutschen Recht nicht so unbedenklich ausschließlich deutschem Recht unterworfen werden können (vgl. Hoeren 2004). Dafür hätte geklärt werden müssen, nach welchem Recht der Lizenzvertrag geschlossen wurde. Da können selbstverständlich zwischen dem Zivilrecht des zuständigen Bundesstaates der USA, dem Zivilrecht Deutschlands oder dem Zivilrecht Chinas bedeutende Unterschiede bestehen. An den deutschen Wohnsitz des Maintainers anzuknüpfen (vgl. Metzger 2004) schlägt dann fehl, wenn er nicht wirklich der Lizenzgeber war. Was ist, wenn die Software von einem taiwanesischen Router-Hersteller von einem taiwanesichem Software-Distributor auf einem Datenträger erworben wurde?

18 Vgl. Hoeren (2004), mit Replik von Metzger (2004) und Kreutzer (2004).

19 Vgl. Spindler (2004, S. 104–106), Zusammenfassung: urheber-, vertrags-, haftungsrechtliche Probleme)]; Hoeren (2004) sieht „zahlreiche Zweifel" an der Entscheidung des LG München I.

Das ist die falsche Betrachtungsweise. OSS ist eine neue, andersgeartete Form des urheberrechtlichen Schaffens. Niemand kann ernsthaft den Wert dieser Art von Leistung und das OSS-Softwareentwicklungsmodell in Frage stellen, daher kann ebenso wenig die Berechtigung zu angemessenem Rechtsschutz in Abrede gestellt werden. Wie auch bei anderen für die deutsche Rechtstradition neuen rechtlichen Phänomenen (wie Leasing, Franchising oder Factoring)[20] muss sich die Rechtsdogmatik an die neue Entwicklung anpassen und kann Neues nicht aussperren. Die Richter des Landgerichts München I haben das verstanden. Denn Richter haben – gegenüber den Rechtsgelehrten – meist den Blick nicht so sehr auf die reine Lehre, sondern mehr für die praktisch vernünftige Lösung.

5.2. Praktische Bedeutung der Entscheidung

Daher kann die Bedeutung der Entscheidung – auch wenn sie nicht die erste zur GPL ist[21] – nicht überschätzt werden und sie wird auch entsprechend aufgenommen.[22]

Sicherlich muss die Bedeutung der Entscheidung in einzelnen Punkten relativiert werden:

- Das Urteil wurde in der ersten Instanz getroffen. Umfassende Rechtssicherheit wird es erst geben, wenn auch Oberlandesgerichte und letztlich der Bundesgerichtshof Entscheidungen zur GPL-Lizenz fällen. Aber diese haben sich mit der vom Landgericht München I getroffenen Entscheidung auseinanderzusetzen und abweichende Ansichten zu begründen, auch wenn sie nicht an die vorliegende Entscheidung gebunden sind. Das vorliegende Verfahren jedenfalls hat seine Beendigung gefunden. Die Entscheidung vom 19.05.2004 ist rechtskräftig, das Unternehmen ist nicht hiergegen vorgegangen.
- Das Urteil ist nicht in einem Hauptsacheverfahren ergangen. Hier entschieden die Richter in einer Eilentscheidung nach Antrag auf Erlass einer einstweiligen Verfügung. Dadurch verliert die Entscheidung nicht an Wert. Der Beschluss ist nicht lediglich nach einseitigem Vortrag des Antragstellers getroffen. Vielmehr wurden nach Widerspruch der Antragsgegnerin beide Seiten gehört. Eine Grundlage für eine ausgewogene Entscheidung war gegeben. Zudem werden häufig die Entscheidungen des Eilverfahrens im Hauptsacheverfahren bestätigt. Das gilt insbesondere dann, wenn über dieselben rechtlichen Fragen zu entscheiden ist und keine neuen Tatsachen hinzukommen. Und gerade dies ist hier der Fall. Für ein Mehr an Rechtssicherheit und damit Normalität ist es wichtig zu wissen, wie Gerichte in einem Fall wie dem vorliegenden entscheiden.

GPL-Software steht demnach nicht im rechtsfreien Raum. Dass hinter der GPL–Software unmittelbar kein Unternehmen steht, macht GPL-Software nicht zum Frei-

20 Metzger (2004) verweist hier zu Recht auf Franchising und Leasing, die ihren Platz im deutschen Recht gefunden haben, obwohl sie zunächst dogmatisch nicht einzuordnen waren.

21 Siehe auch *Progress Software Corp v MySQL AB – U.S. District Court – District of Massachusetts*, Order of February 28, 2002 (http://pacer.mad.uscourts.gov/dc/opinions/saris/pdf/progressf).

22 Trotz kritischer Töne im Ergebnis auch Hoeren (2004).

wild für Unternehmen, die sich gerne mit fremdem Federn schmücken. Die verbleibende Rechtsunsicherheit rührt nicht so sehr aus der unbegründeten Angst, der Großteil der Juristen könnte die GPL-Lizenz in Bausch und Bogen verdammen. Die meisten sind bemüht – wie auch offensichtlich das Gericht – das auf die GPL-Lizenz nicht passende Recht und die Rechtsdogmatik entsprechend anzupassen. Hierbei kann es zukünftig gegebenenfalls noch aus dogmatisch puristischen Erwägungen zu Rückschritten kommen. Hier sind uns die Angelsachsen durchaus überlegen, denen es leichter fällt, die Dinge auch einmal ganz anders zu denken. Die Rechtswelt wird sich jedenfalls dem Erfolg dieses Softwareentwicklungsmodells und dessen Lizenz nicht entziehen können. Und da zwischenzeitlich hinter mancher GPL-Software erhebliche ökonomische Interessen stehen, befördert dies ein Umdenken.

6. Offene Fragen und weitere Risiken – Softwarepatente

Um Missverständnisse zu vermeiden: Das Gericht hat nicht die GPL in Gänze bestätigt, sondern nur einen Teil für das vorliegende Urteil zu prüfen gehabt. Zur GPL gibt es neben den vom Gericht aufgegriffenen noch mehrere offene Fragen. Diese sind aber nebenrangig gegenüber den wirklich ernst zunehmenden Bedrohungen durch Softwarepatente (siehe Abschnitt 6.2., S. 264).

6.1. Offene Fragen zur GPL

Haftung

Ein Gerichtsverfahren ist kein umfassendes Rechtsgutachten zur Wirksamkeit von GPL-Software. Viele Fragen wurden in diesem Verfahren nicht aufgeworfen. Besonders die Frage der Zulässigkeit des Haftungsausschlusses war nicht Gegenstand der rechtlichen Auseinandersetzung. Hier ist mit einer negativen Entscheidung zu rechnen. Das heißt, Entwickler können nicht darauf vertrauen, dass sie keine Haftung für Schäden aus ihrer Software zu tragen haben, nur weil sie für die Softwareüberlassung kein Entgelt erhalten und die GPL-Lizenz einen weitgehenden Haftungsausschluss vorsieht (Ziffer 11 GPL):

> *„11.* There is no warranty for the program, to the extent permitted by applicable law.“

Das Gebot der Stunde ist also weiterhin, Fehler zu vermeiden. Wer die GPL-Software – wie Distributoren oder IT-Systemhäuser – mit weiteren vergütungspflichtigen Leistungen anbietet, unterliegt – zumindest bzgl. des vergütungspflichtigen Teils – ohnehin der üblichen gesetzlichen Haftung und hat sich regelmäßig mit gesonderten Allgemeinen Geschäftsbedingungen abgesichert.

Bereitstellung der GPL und Hinweis auf die GPL

Die Richter haben die Einbeziehung der GPL in das Rechtsverhältnis der Parteien aufgrund der Angaben auf der Website des Softwareprojekts bejaht. Wie bei der

Zurverfügungstellung der Software die GPL bereitgestellt werden muss, streifte das Gericht nur.[23] Wird z. B. bei einem Software-Download-Angebot gar nicht auf die GPL verwiesen und findet man nur durch eine Vielzahl an Clicks zu dem Lizenztext der GPL, kann nach deutschem Recht die Geltung der GPL durchaus in Frage gestellt werden. Man kann OSS-Projekten daher nur anraten, soweit möglich, die GPL aufs Engste mit der Software zu verbinden: also erstens bei Web-Angeboten am Ort des Downloads die GPL im Volltext anzuzeigen oder zumindest einen direkten Link mit sichtbarem Hinweis neben dem Softwaredownload einzurichten sowie zweitens bei automatischen Abruf der Software den Download der Software nur mit dem Lizenztext und entsprechendem Hinweis in einem Downloadpaket zu ermöglichen.

Nicht problematisiert hat das Gericht den Zugriff auf die Software über andere Verbreitungskanäle als den Download der Software von der Website des Entwicklerteams. Der Erwerb der Software kann auch über Distributionen erfolgen oder von nicht mit Hinweisen und GPL-Lizenztext versehenen Servern heruntergeladen werden. Das Unternehmen wäre sicherlich nicht mit der Ausrede davongekommen, man habe die Software „irgendwo im Internet gefunden" und habe dabei keinen Hinweis auf die GPL oder gar den GPL-Lizenztext vorgefunden. Denn es kann unterstellt werden, dass ein Unternehmen, das Software in einem kommerziellen Hardwareprodukt einsetzt und damit einen erheblichen Umsatz erzielt, nicht darauf vertrauen darf, irgendwo Software zu finden und frei nutzen zu dürfen. Spätestens dann wäre das Unternehmen nach angemessener Recherche zwingend auf die Website[24] des Softwareprojekts. gestoßen. Das gilt letztlich für alle – auch nicht kommerzielle – Nutzer. Andererseits sollte man sich seiner Sache als Entwickler und Distributor hier nicht zu sicher sein und zum Wohle von Open Source peinlich darauf achten, dass auch bei anderen Vertriebsformen (u. a. Download von anderen Sites, Distributionsvertrieb) der GPL-Software auf die Geltung der GPL hingewiesen wird und der GPL-Text auch unmittelbar beim Zugänglichmachen der Software zur Verfügung gestellt wird oder in dem Softwarepaket deutlich erkennbar enthalten ist.

Rechtsunsicherheit von OSS? Rechtsunsicherheit von proprietärer Software?

Mit dem Urteil des LG München I wurde belegt, dass Vorwürfe der Rechtsunsicherheit[25] gegenüber OSS und vor allem gegenüber der GPL-Software in weiten Teilen haltlos sind. Die Teilnahme am Rechtsverkehr ist immer mit gewissen Rechtsunsicherheiten verbunden. Das gilt genauso bezüglich OSS wie auch bezüglich proprietärer Software. Nur von den rechtlichen Risiken von OSS zu sprechen (vor allem S. 104–106 Spindler 2003, vgl.), ohne die teilweise höheren Risiken von proprietärer Software dem gegenüber zu stellen,[26] dient der Verunsicherung. Typische Risiken von proprietärer Software sind die Insolvenz des Softwareentwicklers oder die Einstellung des Supports und der Softwarepflege. Ohne Zugriff auf den Quellcode kann eine solche

23 Vgl. auch Kreutzer (2004).
24 http://www.netfilter.org.
25 Vgl. heise newsticker (2003*a*).
26 So nun einschränkend Spindler in Schulz (2003).

Software durch allein eine kleine Änderung wie der Euro-Einführung wertlos werden. Derartige bedeutende Risiken drohen bei OSS nicht.

Ohne Zweifel bestehen noch verschiedene offene Fragen zu OSS. Diese betreffen aber regelmäßig nebenrangige Rechtsfragen oder sind meist in der Praxis anderweitig lösbar. In lizenzrechtlicher Hinsicht werden die Juristen OSS zunächst nicht zu Grabe tragen. Die rechtlichen Gefahren für OSS liegen aber nicht vorrangig im lizenzrechtlichen Bereich. Der Hase liegt ganz woanders im Pfeffer. Auch hier spielt einmal mehr München eine wichtige Rolle.

6.2. Der späte Tod durch Softwarepatente?

Eines der meistbeachtetsten Open-Source-Projekte drohte im Sommer 2004 plötzlich zu scheitern. Die gesamte Verwaltung der Landeshauptstadt München soll auf OSS umgestellt werden. Da platzte im Sommer 2004 die Meldung herein, dass München das Projekt auf Eis legt, weil Softwarepatente eine unkalkulierbare Gefährdung des ca. 30-Millionen-Euro-Projekts darstellen (heise newsticker 2004). Das Projekt wurde gestoppt und eilig ein Gutachten in Auftrag gegeben (vgl. Sedlmaier und Gigerich 2004). Damit wurde das Verletzungsrisiko formal ausgeräumt. Damit ist es aber natürlich nicht getan.

Die entscheidende Bedrohung für GPL-Software ist nicht die Gefahr der fehlenden Wirksamkeit und Durchsetzbarkeit der GPL-Lizenz in Deutschland, wie das vorliegende Gerichtsurteil überzeugend zeigt. Die größte Gefahr geht von den Bestrebungen nach Ausweitung der Patentierbarkeit von softwarebezogenen Erfindungen – „Softwarepatente" – aus (vgl. Ebinger 2000). Diese Gefahr wird oft verniedlicht. Das ist noch nachvollziehbar bei ausgesprochenen Befürwortern des Patentschutzes für Software (vgl. z. B. Wiebe 2004*b*). Relativierende Äußerungen zu den Auswirkungen von Softwarepatenten bei erklärten Vertretern von OSS lösen dagegen Bedenken aus (vgl. etwa Metzger 2000).

Die These, OSS sei in gleicher Weise wie proprietäre Software (*Closed Source*) betroffen, ist schlichtweg falsch. Beim Schutz vor Patenten ist Open Source erheblich benachteiligt wie auch beim Erwerb von Patenten:

- Das Entwicklungsmodell von OSS ist durch Patente existenziell bedroht. Die Entwicklung von OSS setzt die freie Verbreitung und freie Vervielfältigung der Software voraus, ohne die tausende OSS-Entwickler nicht die OSS-Entwicklung sicherstellen können. Diese freie Weiterverbreitung wird durch Patente ausgeschlossen, da jeder zuvor eine Patentlizenz erwerben muss.
- Selbst wenn alle OSS-Entwickler Patentlizenzen erwerben würden, wäre ihnen wohl die Weiterentwicklung der Software – das Kernelement von OSS – verwehrt, da die Entwickler hierzu nach § 11 Nr. 2 PatG nicht berechtigt sind.
- Der offene Entwicklungsprozess von OSS erlaubt keine Patentierung, da eine an die Öffentlichkeit gelangte Erfindung grundsätzlich nicht mehr patentiert werden kann.

– Ein Softwareunternehmen kann sich die Ausgaben für (aussichtslose) Patentprozesse leisten. Bei OSS sind die finanziellen Reserven der Entwickler schnell ausgeschöpft. Regelmäßig werden OSS-Entwickler bei dem Vorwurf einer Patentverletzung wegen des ungewissen Ausgangs eines Patentrechtsstreits die Softwareentwicklung einstellen (vgl. atai 2000) oder selbst wenn sie Recht haben wegen der hohen Kosten eines Patentverletzungsprozesses gleichwohl klein beigeben. Zudem werden allein aus Angst vor Patentrechtsstreitigkeiten bestimmte OSS-Projekte von Entwicklern gar nicht erst angegangen.

– Die Ausweitung der Vergabe von Softwarepatenten hat dazu geführt, dass viele Unternehmen sich nur deshalb Patente sichern, um für den Fall einer Verletzung fremder Patente sich mit eigenen Patenten verteidigen zu können (Rüstungsspirale und Kreuzlizenzierung). Voraussetzung hierfür sind aber eigene Patente. Hier scheitert es bei OSS nicht nur an rechtlichen Hindernissen, sondern auch an den fehlenden finanziellen Mitteln für den Patenterwerb. Für die Anmeldung und Aufrechterhaltung nur eines Patents in den wesentlichen Softwaremärkten sind mehrere Zehntausende Euro zu veranschlagen. Hierfür sind bei den Open-Source-Entwicklern keine Ressourcen vorhanden.

– Ob eine Software ein Patent verletzt, kann bei OSS bedeutend leichter festgestellt werden als bei proprietärer Software. Denn der Quellcode von proprietärer Software ist in den allermeisten Fällen nicht von Dritten einsehbar. Reverse Engineering des ausführbaren Objektcodes ist regelmäßig zu aufwendig, jedenfalls für OSS-Entwickler. Auf die Gefährdung von OSS durch Softwarepatente ist vielfach hingewiesen worden (vgl. Lutterbeck et al. 2000, Schölch 2001). Das Damokles-Schwert der Softwarepatente hängt nur an einem dünnen Faden über OSS. Es werden keine wirksamen Lösungen angeboten, die die Konfliktlage effektiv beseitigen könnten. Dies gestehen selbst Softwarepatentbefürworter ein (vgl. Wiebe 2004*a*,*b*).

– Soweit auf die Registrierung von Open-Source-Entwicklungen verwiesen wird, so entstehen dadurch nur weitere Kosten. Zudem schützt dies nur vor der Patentierung von Ideen, die bereits zuvor (als Open Source) veröffentlicht wurden.

– Softwarepatente setzen voraus, dass die Entwickler vor Softwareerstellung sehr kostspielige Patentrecherchen ausführen, ob das geplante Projekt keine Patente verletzt. Diese Kosten können Open-Source-Projekte nicht aufbringen.

– Sicher gibt es auch Open-Source-freundliche Unternehmen, die über Patente verfügen. Doch zunächst werden diese ihre eigenen Produkte schützen. Doch wird der Schutz auch auf die freien Entwickler und gar auf die Wettbewerber erstreckt werden? Wenn man Open Source nur noch nützen kann, wenn man Produkte (Hard- oder Software) eines Unternehmens kauft oder sich von Patentprozessen durch eine spezielle Vereinbarung freikauft, dann wird das Grundmodell von Open Source nicht mehr funktionieren. Denn nur der freie Wettbewerb hat Open Source groß gemacht. Die beste Idee setzt sich durch und nicht das Beharren auf ein Monopolrecht.

Für die unbegründete Hoffnung, dass Kompatibilitätsprobleme zwischen OSS und Softwarepatenten eher in geringerem Maße auftreten würden (vgl. Wiebe 2004*a*, S. 265, Rn. 89), spricht nichts. Bei einer weiter zunehmenden wirtschaftlichen Bedeutung von OSS und den hohen Kosten der Patentinhaber für die Patenterlangung spricht alles dafür, dass die Patente auch gegenüber OSS eingesetzt werden. Für eine Verbesserung der Qualität der erteilten softwarebezogenen Patente spricht auch nichts. Die über Jahre bestehende Kritik hat keine Verbesserung gebracht. Die zunehmenden Einsparungen bei den Patentämtern wird tatsächlich zu einer zusätzlichen Verschlechterung der Recherche und Erteilungsentscheidung führen. Wurde aber ein Patent zu Unrecht erteilt und hieraus OSS oder auch kleinen und mittelständische Unternehmen (KMUs) eine Patentverletzung vorgeworfen, werden diese aus finanziellen Gründen die hohen Kosten einer Patentstreitigkeit scheuen (so auch Wiebe 2004*a*, S. 257, Rn. 69) und sich dem Patentinhaber beugen. Die Dokumentation des Standes der Technik im Bereich der Softwareentwicklung (dazu Wiebe 2004*b*, S. 292) ist Wunschdenken, wenn gleichzeitig festgestellt wird, dass jede veröffentlichte OSS neuen Stand der Technik schaffen kann (wie bei Wiebe 2004*b*, S. 289). Eine solche Dokumentation wäre eine Sisyphus-Arbeit. Es versteht sich von selbst, dass die Patentprüfer selbstverständlich den so veröffentlichten Stand der Technik nicht zur Kenntnis nehmen können und daher in Unkenntnis des wahren Standes der Technik Patente erteilen.

Es gibt daher nur zwei Auswege:

- Genereller Verzicht auf softwarebezogene Erfindungen/Softwarepatente: da selbst deren Befürworter deren wirtschaftliche Sinnhaftigkeit in Frage stellen (vgl. Wiebe 2004*b*, S. 279); oder
- Keine Erstreckung von Patenten auf Open-Source-Software: Denn OSS erfüllt eine Informationsfunktion, die auch dem Patentwesen zugrunde liegt (Wiebe 2004*a*, S. 246, Rn. 44), indem der Stand der Softwareentwicklung durch den offenen Quellcode jedermann zur Nutzung und Weiterentwicklung zur Verfügung gestellt wird. Hierdurch erfüllen die OSS-Entwickler mit höherem volkswirtschaftlichem Nutzen kostenlos die Funktion eines kostspieligen Patentamtes (OSS, die „alternative Patentdatenbank"). Zwei Patentämter können nicht nebeneinander im selben Gebiet konkurrieren. Sofern nicht auf die Erteilung von staatlichen Monopolrechten für Softwarepatente verzichtet wird, so muss für OSS als „alternatives Patentamt" eine Bereichsausnahme gelten. Der Schutz von Patenten ist daher nicht auf OSS zu erstrecken (Ebinger 2001).

Die EU hatte angekündigt, die Interessen von Open Source ausreichend berücksichtigen zu wollen (vgl. u. a. Kommission der Europäischen Gemeinschaften 2000, S. 3), wie die Rechtsliteratur zeigt, ist dies nicht möglich. Die EU und die hinter ihr stehenden Mitgliedstaaten haben nun die Möglichkeit, ihren Worten Taten folgen zu lassen.

Literaturverzeichnis

DiBona, C., Ockman, S. und Stone, M. (Hrsg.) (1999), *Open Sources. Voices from the Open Source Revolution*, O'Reilly, Sebastopol, CA. http://www.gnu.org/gnu/thegnuproject.html [28. Jan 2005].

Ebinger, T. (2000), 'Höchstrichterliche Umwege – Bundesgerichtshof ebnet den Weg für Softwarepatente', *c't Magazin für Computertechnik* **17**(25), S. 264.

Ebinger, T. (2001), 'Grenzen des patentrechtlichen Schutzes von Computerprogrammen unter besonderer Berücksichtigung von nach der GNU GPL lizenzierten Computerprogrammen'. unveröffentlicht.

Ebinger, T. (2002), 'Open Source Software – eine Einführung', *Der Syndikus (Informationstechnologie/Neue Medien)* (März/April). http://www.der-syndikus.de/frames/synd_fr.htm [18. Nov 04].

Ebinger, T. (2003), 'Rechtliches Minenfeld? Umstrittene Studie zur Rechtssicherheit von Open Source', *c't Magazin für Computertechnik* **20**(15), S. 44.

Free Software Foundation Europe (2003), 'VSI-Studie mit gravierenden fachlichen Mängeln?', http://www.germany.fsfeurope.org/de/news/2003/vsi-studie.html, 3. Jul 2003 [28. Jan 2004].

Gehring, R. A. (1996), 'Freeware, Shareware und Public Domain', http://ig.cs.tu-berlin.de/oldstatic/ap/rg/1996-06/GehringRA-FreewareSharewarePublicDomain-13061996.pdf [13. Jan 2005]. Studienarbeit, TU Berlin, Informatik und Gesellschaft.

Gehring, R. A. und Lutterbeck, B. (Hrsg.) (2004), *Open Source Jahrbuch 2004*, Lehmanns Media, Berlin.

Hoeren, T. (2004), 'Wirksamkeit einer GPL-Lizenz – LG München I, Urteil vom 19.5.2004, mit Anmerkungen von Hoeren und Metzger', *Computer und Recht* (10), S. 774–779.

Koch, F. A. (2000*a*), 'Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software, Teil 1', *Computer und Recht* (5), S. 273–281.

Koch, F. A. (2000*b*), 'Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software, Teil 2', *Computer und Recht* (6), S. 333–344.

Kommission der Europäischen Gemeinschaften (2000), 'Die Patentierbarkeit computer-implementierter Erfindungen (Sondierungspapier der Dienststellen der Generaldirektion Binnenmarkt)', http://europa.eu.int/comm/internal_market/en/indprop/comp/softde.pdf, 19.10.2000 [28. Jan 2005].

Kreutzer, T. (2004), 'LG München I: Wirksamkeit der GNU General Public Licence (GPL) nach deutschem Recht', *Multimedia und Recht* (10), S. 693–694.

Landgericht München (2004), 'Urteil des Landgerichts München vom 19.05.2004 – Aktenzeichen: 21 O 6123/04', http://www.jbb.de/urteil{_}lg{_}muenchen{_}gpl.pdf [07. Feb 2005].

Linux Verband (2003), 'Software-Verband nutzt Rechtsstudie zu Open Source fürs Lobbying', http://linux-verband.de/nachrichten/allgemein/20030703111952, 03.07.2003 [14. Jan 2005].

Lutterbeck, B., Horns, A. H. und Gehring, R. A. (2000), 'Sicherheit in der Informationstechnologie und Patentschutz für Softwareprodukte – ein Widerspruch? (Kurzgutachten für den Bundeswirtschaftsminister)', http://ig.cs.tu-berlin.de/forschung/IPR/LutterbeckHornsGehring-KurzgutachtenSoftwarePatente-122000.pdf [28. Jan 2005].

Metzger, A. (2000), 'Softwarepatente als Gefahr?', *Linux Magazin* (7), S. 64–65. http://www.ifross.de/ifross_html/art4.html [28. Jan 2005].

Metzger, A. (2004), 'Wirksamkeit einer GPL-Lizenz – LG München I, Urteil vom 19.5.2004, mit Anmerkungen von Hoeren und Metzger', *Computer und Recht* (10), S. 774–779.

Metzger, A. und Jaeger, T. (1999), 'Open Source Software', *GRUR Int.* S. 839 ff.

Müller-Broich, J. D. (1998), *Autodistributive Computersoftware. Shareware, Freeware, und Public Domain Software als Sonderform der Softwareüberlassung*, Peter Lang Verlag, Frankfurt am Main. zugl. Dissertation.

Open Source Initiative (2004*a*), 'History of the OSI', http://www.opensource.org/docs/history.html.

Open Source Initiative (2004*b*), 'Open Source Definition', http://www.opensource.org/docs/definition.php.

Raymond, E. S. (1998), 'The Cathedral and the Bazaar', *First Monday* **3**(3). http://firstmonday.org/issues/issue3_3/raymond/index.html [28. Jan 2003].

Schölch, G. (2001), 'Softwarepatente ohne Grenzen', *GRUR* (1), S. 16–21.

Schulz, C. (2003), 'Die scharfe Klinge des Gesetzes?', *Linux Magazin* (9), S. 68–70. http://www.ifross.de/ifross_html/art35.pdf [28. Jan 2005].

Sedlmaier, R. und Gigerich, J. (2004), 'Rechtliche Bedingungen und Risiken der Landeshauptstadt München für den Einsatz von Open-Source Software (Kurzgutachten)', http://www.ris-muenchen.de/RII/RII/DOK/SITZUNGSVORLAGE/517379.pdf, 10. Sep 2004 [28. Jan 2005].

Sester, P. (2000), 'Open-Source-Software: Vertragsrecht, Haftungsrisiken und IPR-Fragen', *Computer und Recht* (12), S. 797 ff.

Siepmann, J. (1999), 'Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software', *JurPC Web-Dok* (163). http://www.jurpc.de/aufsatz/19990163.htm [18. Nov 2003].

Spindler, G. (2003), 'Rechtsfragen der Open Source Software', Gutachten im Auftrage des Verbandes der Softwareindustrie in Deutschland e.V., http://www.vsi.de/inhalte/aktuell/studie_final_safe.pdf, 03.07.2003 [18. Nov 2004].

Spindler, G. (Hrsg.) (2004), *rechtsfragen bei open source*, Verlag Dr. Otto Schmidt, Köln.

Stallman, R. (1996), 'Free Software Foundation: What is Copyleft?', http://www.gnu.org/copyleft/copyleft.html [18. Nov 2004].

Stallman, R. (1999), *The GNU Operating System and the Free Software Movement*, in DiBona et al. (1999), S. 53–70. http://www.gnu.org/gnu/thegnuproject.html [28. Jan 2005].

Wiebe, A. (2004*a*), *Patentrecht*, in Spindler (2004), S. 223–266.

Wiebe, A. (2004*b*), *Patentschutz und Softwareentwicklung – ein unüberbrückbarer Gegensatz?*, in Gehring und Lutterbeck (2004), S. 277–304.

atai (2000), 'Microsoft patents ASF media file format, stops reverse engineering', http://www.advogato.org/article/101.html, 5. Jun 2000 [28. Jan 2005].

heise newsticker (2001), 'IBM kritisiert Microsofts Open-Source-Kritik', http://www.heise.de/newsticker/meldung/19058, 06.07.2001 [18. Nov 2004].

heise newsticker (2003*a*), 'Software-Verband nutzt Rechtsstudie zu Open Source fürs Lobbying', http://www.heise.de/newsticker/meldung/print/38706, 18.07.2003 [12. Jan 2005].

heise newsticker (2003*b*), 'VSI weiter unter Beschuss wegen Linux-Rechtsstudie', http://www.heise.de/newsticker/meldung/print/38290, 04.07.2003 [12. Jan 2005].

heise newsticker (2004), 'München legt Linux-Projekt wegen der Softwarepatente auf Eis', http://www.heise.de/newsticker/meldung/print/49735, 04.08.2004 [28. Jan 2005].

Kapitel 5

Gesellschaft

„Everything that is really great and inspiring is created by the individual who can labor in freedom."

– Albert Einstein (1950)

Open Source – Zwischen Geschichte und Zukunft

Sebastian Ziebell

1. Einleitung zum Kapitel „Gesellschaft“

Setzt man sich eingehender mit Open Source auseinander, erkennt man schnell, wie vielschichtig und komplex deren Strukturen sind. Es scheint sich, oberflächlich betrachtet, in ähnlicher Weise wie in der freien Marktwirtschaft zu verhalten,[1] in der jeder einzelne Mitwirkende zuerst an sein eigenes Wohl denkt, davon aber alle insgesamt profitieren. Die Zahl der Open-Source-Projekte und deren Teilnehmer ist dabei in den letzten Jahren kontinuierlich angewachsen, wie man z. B. den verschiedenen Plattformen wie „sourceforge“[2] entnehmen kann. Es handelt sich offensichtlich um mehr, als nur eine Alternative innerhalb der Softwareentwicklung. Das Fundament, auf dem Open Source basiert, liegt in einem anderen Verständnis des Eigentumbegriffs begründet. Das geschaffene Wissen wird der Allgemeinheit übertragen, jeder hat Zugriff darauf. Niemand soll auf diese Weise von erzeugtem Wissen ausgeschlossen werden können. Vielmehr soll durch Kooperation und Teilnahme bestehendes Wissen genutzt werden, um Ideen und Innovationen und damit neues und besseres Wissen schaffen zu können.

Dieses Kapitel versucht, einen Einblick in das Phänomen Open Source aus der Perspektive der Geisteswissenschaften zu gewähren. Die hier vorgestellten Beiträge sollen dazu dienen, den Blick für weitere kulturelle, historische und gesellschaftliche Überlegungen und Fragen zu schärfen. Nicht selten sind allerdings Überlegungen und Theorien, die sich mit dem Phänomen Open Source aus einem gesellschaftlich, kulturellen Blickwinkel befassen, pathetisch angehaucht oder empirisch nicht unbedingt gehaltvoll. In der Redaktion gab es einige kontroverse Diskussionen bezüglich solcher Texte. Sie stellen jedoch den jetzigen Diskussionsstand dar, daher haben sich die Redaktion und die Herausgeber für das Abdrucken der Texte entschieden, wenn sie die Relevanz für weitere Diskussionen als gegeben ansahen. Jeder Leser sei hier

1 Adam Smith beschrieb 1776 in seinem Werk „An Inquiry into the Nature and Causes of the Wealth of Nations“ das Grundprinzip der freien Marktwirtschaft. Teil dieser freien Marktwirtschaft ist eine Selbstregulierung durch die sog. *unsichtbare Hand*, nach der jeder Mensch eigennützig handelt und so zum Wohl der gesamten Gesellschaft beiträgt.

2 Siehe http://sourceforge.net/.

dazu angehalten, die betrachteten Überlegungen zu reflektieren und sich selbst ein Urteil über die Qualität der Beiträge zu bilden. Denn es treffen unterschiedliche Ansichten und Vorstellungen aufeinander, die in vielen Fällen mehr Fragen aufwerfen, als klärende Antworten geben zu können.

2. Eigentum verpflichtet

Der grundlegende Unterschied zwischen Open-Source-Software und proprietärer Software liegt in den differierenden Auffassungen des Eigentumverständnisses und der damit verbundenen Rechte. Um diese Differenz zu verdeutlichen, soll kurz der Begriff des Eigentums beschrieben werden. Die damit verbundenen Eigentumsrechte, insbesondere bei „geistigem Eigentum", zeigen, welche rechtlichen Schutzinstrumente gegeben sind, um die verschiedenen Auffassungen durchzusetzen. Diese Ausführungen sollen verständlich machen, wie diese Instrumente funktionieren um sowohl die Interessen von Urhebern kreativer und innovativer Werke zu vertreten als auch das Interesse der Öffentlichkeit. Zum einen sollen Anreize geboten werden, überhaupt Innovationen, Ideen und Wissen zu erzeugen. Diese Anreize werden durch die Schutzinstrumente gesichert. Auf der anderen Seite besteht das Interesse der Allgemeinheit daran, die getätigten geistigen Werke für weitere Prozesse verwenden zu können, um den Wohlstand aller zu mehren. Die Schutzinstrumente müssen so geregelt sein, dass sich durch diese ein Gleichgewicht bildet, welches beiden Interessensseiten zugute kommt.

2.1. Materielles Eigentum

John Locke hat 1690 in seiner Abhandlung „Über die Regierung" den Begriff des Eigentums als angeborenes natürliches Recht erklärt, wonach jeder Mensch das Eigentum an einer Sache erwirbt, in dem er sich diese durch Arbeit zu Eigen macht. Darin enthalten war das Recht, andere von dieser Sache auszuschließen (Euchner 1992, Kapitel 5, Absatz 26). Diese Verständlichkeit des Begriffs wurde bis heute bewahrt und ist so von der Gesellschaft übernommen worden. Dem Eigentum kommt zudem ein nahezu übergesetzliches Recht zu, welches niemandem genommen werden kann, nicht einmal vom Staat.[3] Eigentum beschreibt das Verhältnis zwischen Menschen bezüglich einer Sache, nicht die Sache selbst (Heller und Nuss 2003, S. 397). Eigentum an einer Sache ist etwas anderes als der Besitz. Während man bei Eigentum über die Nutzungs- und Verwertungsrechte an einer Sache verfüge, beschreibt der Besitz das tatsächliche Innehaben dieser Sache. So kann man als Eigentümer beispielsweise seine Zeitung einem Freund ausleihen, der dadurch zum Besitzer wird. Somit ist einem das Recht vorbehalten andere an der Nutzung auszuschließen, also andere am Lesen der Zeitung zu hindern. Leider ist ein Dieb, der einem die Zeitung entreißt, ebenso im

3 Dies kann in Ausnahmefällen geschehen, muss dazu aber begründet sein und dem Allgemeinwohl dienen: „Eine Enteignung ist nur zum Wohle der Allgemeinheit zulässig. Sie darf nur durch Gesetz oder auf Grund eines Gesetzes erfolgen, das Art und Ausmaß der Entschädigung regelt" (Grundgesetz, Artikel 14, Absatz 3).

Besitz derselben. Es besteht ferner die Möglichkeit die Eigentumsrechte auf andere zu übertragen. Würde man die Zeitung seinem Freund schenken, könnte dieser von nun ab über die Nutzungsrechte entscheiden und regeln, wer die Zeitung lesen darf und wer nicht. Der Begriff Besitz wird meist synonym für Eigentum verwendet, was unter Umständen, insbesondere im juristischen Sinn, zu Missinterpretationen führen kann. In diesem Kontext sind insbesondere materielle Güter gemeint, die sich physisch greifen lassen.

Bei materiellen Gütern handelt es sich um Produkte, die meist knapp sind, also nur in endlicher Zahl vorhanden sind oder produziert werden. Zum einen können die Ressourcen begrenzt sein (z. B. Erdöl, Kohle) oder sie werden künstlich knapp gehalten. Einfach betrachtet wird aus Angebot und Nachfrage der Preis ermittelt. Die Kosten für materielle Güter betreffen jedes Produkt gleichermaßen.

2.2. Immaterielle Güter und „geistiges Eigentum"

Eine andere Kategorie von Eigentum bilden die immateriellen Güter.[4] Sie sind als solche nicht physisch erfassbar. So sind Ideen, Theorien, kreative und intellektuelle Prozesse Werke des Geistes und bilden Produkte immaterieller Art. Die bekanntere Bezeichnung für immaterielle Güter ist „geistiges Eigentum". Im anglo-amerikanischen Raum wird, mit einigen Unterschieden, der Begriff *intellectual property* verwendet. Bei materiellen Gütern sind die Zuweisung, die Verwendung und die Übertragung in den Rechtssystemen geregelt (Grassmuck 2004, S. 48). Für immaterielle Güter galt dies lange Zeit nicht, da sich die Rechtssysteme schwer damit taten, Eigentumsrechte bei immateriellen Gütern ähnlich zu handhaben wie bei materiellen Gütern. In Deutschland konnte dies erst Mitte des vorigen Jahrhunderts erreicht werden. Dies schloss auch die Möglichkeit ein, ähnlich wie bei materiellen Gütern die Eigentumsrechte auf andere zu übertragen, also die Nutzungsrechte an einem Werk vollständig weiterzugeben. Hier unterscheidet sich das kontinental-europäische Urheberrecht vom anglo-amerikanischen Copyright dadurch, dass der Urheber eines Werkes sehr eng mit seinem Werk verbunden bleibt, auch nachdem dieser die vollständigen Nutzungsrechte übertragen hat, wohingegen beim Copyright die Rechte meist vollständig an einen Verwerter übertragen werden (Grassmuck 2004, S. 59).

Immaterielle Güter besitzen andere Eigenschaften als materielle Güter. Ein Verbrauch, wie er bei materiellen Gütern der Fall ist, tritt bei immateriellen Gütern nicht ein. Ein Buch, welches man liest, ist nach dem Lesen nicht verbraucht. Die Abnutzungs- und Gebrauchsspuren betreffen höchstens das Medium: in diesem Fall die Seiten des Buches, nicht aber das geistige Werk, welches als Text abgedruckt ist. Ebenso verhält es sich mit Ideen, die man gegenüber einem anderen äußert. Dadurch das man sie weitergibt, verliert man sie nicht, vielmehr kann der andere Nutzen daraus ziehen oder eben nicht. Die Grenzkosten für diese Güter fallen niedriger aus, da die Kosten nur für das Ursprungsgut zu leisten sind. Solange die Distribution von „geistigem Eigentum" an Medien (z. B. Bücher, Schallplatten) gebunden war, mit der

4 Aus dem ersten französischen Urheberrechtsgesetz 1791: „Es handelt sich um eine Art von Eigentum, die sich gänzlich von anderen Arten des Eigentums unterscheidet" (Vgl. Grassmuck 2004, S. 48).

eine ähnliche Knappheit erreicht werden konnte wie bei materiellen Gütern, war eine vergleichbare Verwertbarkeit gegeben.

2.3. Eigentumsrechte mit Blick auf „geistiges Eigentum“

Da man „geistiges Eigentum“ auf eine ähnliche Weise wie materielle Güter verwertbar machen wollte, waren bestimmte Mechanismen nötig, die dies gewährleisten sollten. Mit dem „geistigen Eigentum“ sind die staatlich gesicherten Schutzinstrumente Patentschutz, Urheberschutz und Marken-/Musterschutz verknüpft. Sie räumen dem Urheber geistigen Eigentums über einen temporären Zeitraum exklusive Verwertungs- und Nutzungsrechte ein. Innerhalb dieses Zeitraumes erhält der Schöpfer ein künstliches Monopol über sein Werk. Diese Rechte sind in ähnlicher Weise wie die Eigentumsrechte bei materiellen Gütern auf andere übertragbar. Somit kann der Urheber sein Werk an einen Produzenten verkaufen, der danach über die Verwertung und Nutzung verfügt.

Ein wesentlicher Unterschied zwischen Rechten an materiellen und immateriellen Gütern besteht darin, dass man bei Erwerb letzterer, nur im Besitz darüber ist und nicht über die Eigentumsrechte daran verfügt. Dies ist durch die Schutzmechanismen geregelt. Die exklusiven Verwertungs- und Nutzungsrechte liegen beim Urheber. Bei einem Buch verfügt man über die Eigentumsrechte am Medium, ist in dieser Form aber nur Besitzer des Inhaltes. So kann man zwar andere am Lesen seines Exemplars ausschließen, aber niemandem davon abhalten sich dieses Buch selbst zuzulegen, um den Inhalt zu lesen.

Der Zweck dieser Instrumente ist es, dem Schöpfer die Möglichkeit zu bieten, aus seiner getätigten geistigen Leistung Profit zu erwirtschaften. Nach Ablauf der exklusiven Verwertbarkeitsdauer geht so erschaffenes „geistiges Eigentum“ an die Allgemeinheit über. Die Eigentumsrechte verfallen und das Werk wird zum Gemeingut. So ist beispielsweise beim deutschen Urheberrecht die Frist so bemessen, dass der Schutz 70 Jahre nach dem Tod des Urhebers verfällt. Die Allgemeinheit hat nach Ablauf der Fristen die Möglichkeit, diese Werke zu nutzen und darauf aufbauend neue Innovationen und Werke hervorzubringen. Im nachfolgenden Kapitel „Open Content“ werden alternative Möglichkeiten gezeigt, wie man kreative und schöpferische Werke der Allgemeinheit zur Verfügung stellen kann, ohne auf alle Rechte verzichten zu müssen.

Die Schutzinstrumente bei „geistigem Eigentum“ fungieren als Anreize, Profit aus seiner kreativen und schöpferischen Leistung zu erzielen. Innovation soll auf diese Weise gefördert werden. Auf der anderen Seite sollen die Verwertbarkeitszeiten so bemessen sein, dass die Allgemeinheit nach Ablauf der Fristen Zugang zu den Werken erhält und sie für weitere kulturelle, gesellschaftliche und innovative Prozesse nutzen kann. Eine auf den Wohlstand der Allgemeinheit ausgerichtete Gesellschaft muss daher zum Ziel haben, das optimale Gleichgewicht zwischen dem privaten und öffentlichen Interesse zu finden (Heller und Nuss 2003, S. 397; Lessig 2001, S. 97–99).

So findet sich diese Forderung unter anderem in der Erklärung der Menschenrechte

der Vereinten Nationen[5] und im deutschen Grundgesetz wieder, wo es heißt: „Eigentum verpflichtet. Sein Gebrauch soll zugleich dem Wohle der Allgemeinheit dienen" (Grundgesetz, Artikel 14, Absatz 2). Sind die Zeiten für eine exklusive Verwertbarkeit der Eigentumsrechte zu kurz, bietet man womöglich zu geringe Anreize, um Wissen, Ideen und Innovationen hervorzubringen. Die Allgemeinheit wäre zwar frühzeitig in der Lage, von den getätigten Werken zu profitieren. Durch den Mangel an Innovationen wäre die Schaffung weiterer Innovationen und Werke, die darauf aufbauten, jedoch nicht gegeben. Sind die Schutzfristen dagegen sehr lang bemessen, wären die Anreize für die Schaffung von Innovationen zwar höher, nur auch die Allgemeinheit dann länger von der Nutzung ausgeschlossen. Um das richtige Gleichgewicht zwischen dem Interesse der Öffentlichkeit und den Besitzern von urheberrechtsgeschütztem Material zu gewährleisten, müssen die Schutzmechanismen stets im kulturellen, gesellschaftlichen und technologischen Kontext neu gefunden werden (siehe Heller und Nuss 2003).

2.4. Gleichgewicht der Interessen

Solange „geistiges Eigentum" an Medien wie Bücher oder Schallplatten gebunden war, die eine Vervielfältigung und eine Verbreitung des Werkes nicht ohne zusätzliche Kosten zuließen, verhielten sich die Nutzung und Verwertung ähnlich wie bei materiellen Gütern. Das Gleichgewicht zwischen den Interessen, welches durch die Schutzinstrumente resultierte, war weitestgehend gegeben. Als die Digitalisierung dieser immateriellen Güter eintrat, blieben die Werke nicht mehr ans Medium gebunden. Durch die schnelle technologische Entwicklung von Computern in den letzten Jahren, insbesondere die des Internets, sind die Möglichkeiten für eine Vervielfältigung und Verbreitung „geistigen Eigentums" in digitaler Form in höchstem Maße gegeben. Mit der Digitalisierung treten neue Eigenschaften auf. Eine Kopie unterscheidet sich in keiner Weise vom Original. Ein weiterer Nutzer schränkt den Nutzungsraum für andere nicht ein. Das Werk wird nicht verbraucht, wenn man es nutzt. „Geistiges Eigentum" in digitaler Form wird anders wahrgenommen als in analoger Form, da die Konsumption eine andere ist.

In den letzten Jahren wurde versucht, den Konsequenzen dieser Digitalisierung mit restriktiveren Gesetzesregelungen entgegenzuwirken. So wurde beispielsweise im September 2003 das Urheberrecht hierzulande angepasst.[6] Diese Anpassung sieht vor, dass digitale Güter, die über ein wirksames Kopierschutzverfahren verfügen, nicht mehr kopiert werden dürfen. Auf diese Weise gekapseltes „geistiges Eigentum" darf auch für private Zwecke nicht kopiert werden. In der nächsten Novellierung

5 „(1) Jeder hat das Recht, am kulturellen Leben der Gemeinschaft frei teilzunehmen, sich an den Künsten zu erfreuen und am wissenschaftlichen Fortschritt und dessen Errungenschaften teilzuhaben.
(2) Jeder hat das Recht auf Schutz der geistigen und materiellen Interessen, die ihm als Urheber von Werken der Wissenschaft, Literatur oder Kunst erwachsen" (Vgl. Grassmuck 2004).

6 Siehe hierfür die Presseerklärung vom 12. September 2003 des Bundesministerium der Justiz (http://www.bmj.bund.de/enid/58.html?presseartikel_id=222) und das Bundesgesetzblatt Nr. 46 (http://www.bmj.bund.de/files/9d2345abeb7d864e53d01f444e3cabe4/441/Bundesgesetzblatt_nr46.pdf).

soll das Recht auf Privatkopie sogar gänzlich gestrichen werden.[7] Dies hat auch Konsequenzen für die Nutzung von Werken, deren Schutzfristen abgelaufen sind. So darf man den Inhalt nicht für eigene Zwecke verwenden, wenn das Medium mit einem wirksamen Kopierschutzverfahren versehen ist. Somit wird der vormals zulässige Handlungsfreiraum eingeschränkt. Auch in anderen Bereichen soll der Schutz auf „geistiges Eigentum" ausgeweitet werden. Wie in der Einleitung des Kapitels „Recht und Politik" von Katja Luther beschrieben, zeigt die derzeit heftig geführte Debatte um Softwarepatente in der EU, wie schwierig es ist, eine Entscheidung zu finden, die die Interessen beider Seiten berücksichtigt. Mit diesen angepassten Schutzrechten soll der veränderte Umgang mit „geistigem Eigentum" in digitaler Form wieder in das alte Verwertungsschema gerückt werden.

In den USA wurden in den letzten Jahrzehnten zahlreiche Änderungen an den Rechten „geistigen Eigentums" vorgenommen.[8] Insbesondere durch den Digital Millenium Copyright Act (DMCA) und durch Softwarepatente wurde versucht, auf die technologischen Entwicklungen zu reagieren. Allerdings veränderte sich damit auch das bisherige Gleichgewicht zwischen den Interessen der Rechteinhaber und denen der Öffentlichkeit. Die Rechte der Inhaber wurden ausgeweitet und ihnen wurde eine stärkere Kontrolle ermöglicht, während die Rechte der Allgemeinheit eingeschränkt wurden.[9]

So sieht beispielsweise der Kultur- und Kommunikationswissenschaftler Siva Vaidhyanathan ein Grundrecht des amerikanischen Copyrights außer Kraft gesetzt: „Fair use, while not quite dead, is dying" (2003).[10] Er führt an, wie mit dem Hinweis auf den DMCA kritische oder satirische Beiträge, die unter Beachtung des Fair-Use-Prinzips urheberrechtliches Material verwendeten, durch Internetanbieter zensiert oder gefiltert wurden.

3. Open Source als öffentliches Gut

Zwischen den Herstellern von proprietärer Software und Open-Source-Software bestehen unterschiedliche Auffassungen zum Eigentumsverständnis. Bei ersteren liegen die Eigentumsrechte vollständig beim Produzenten, d. h. als Besitzer dieser Software verfügt man nur über eingeschränkte Nutzungsmöglichkeiten. Bei Open Source werden diese Rechte mit bestimmten Bedingungen an die Allgemeinheit übertragen. Die

7 Siehe das eingerichtete Forum des BMJ (http://www.kopien-brauchen-originale.de).

8 „These extensions are relatively new. In the first hundred years, Congress retrospectively extended the term of copyright once. In the next fifty years, it extended the term once again. But in the last forty years, Congress has extended the term of copyright retrospectively eleven times" (Lessig 2001, S. 107).

9 „These balances, however, are not, on balance, even: though limits have been drawn, the net effect is increased control. The unavoidable conclusion about changes in the scope of copyright's protections is that the extent of „free content" – meaning content that is not controlled by an exclusive right – has never been as limited as it is today. More content is controlled by law today than ever in our past" (Lessig 2001, S. 110).

10 „Als Fair Use bezeichnet man eine Rechtsdoktrin des angloamerikanischen Copyright-Systems, die bestimmte nicht autorisierte Nutzungen von geschütztem Material zugesteht, sofern sie der öffentlichen Bildung und der Anregung geistiger Produktionen dienen; sie erfüllt eine vergleichbare Funktion wie die Schrankenbestimmungen des kontinentaleuropäischen Urheberrechts." (Wikipedia)

Weitergabe, Vervielfältigung und Modifikation ist gegeben. Beiden gemein ist, dass die Nutzungs- und Verwertungsrechte in Form von Lizenzen vereinbart sind. Die einzelnen Open-Source-Lizenzen unterscheiden sich in Bezug auf die weitere Nutzung des Quelltextes. So setzt die GPL voraus, dass Modifikationen wieder unter dieser Lizenz veröffentlicht werden müssen. Andere Lizenzen, wie z. B. die BSD-Lizenz, ermöglichen es, den Quelltext in proprietärer Software zu nutzen. So liegen den Open-Source-Lizenzen verschiedene Auffassungen zugrunde, wie das Wissen weiter genutzt werden darf.

Die wichtigste Forderung von Open Source ist es, dass niemand davon ausgeschlossen werden darf, die Software zu vervielfältigen, weiterzugeben und zu modifizieren. Erfüllt eine Lizenz diese Voraussetzung, wird ihr von der *Open Source Initiative* ein Zertifikat verliehen. Jeder hat die Möglichkeit, Open-Source-Software zu nutzen, niemand wird am Zugang gehindert. Durch Open Source wird das erschaffene Wissen innerhalb der verschiedenen Projekte genutzt. Mit vorhandenem Wissen wird neues geschaffen, bestehendes kann modifiziert und weiterentwickelt werden. So wird das Rad nicht ständig neu erfunden, sondern dort, wo Bedarf besteht, wird entwickelt. Dabei ist die so geschaffene Software für jeden frei zugänglich und nutzbar. So weist Open-Source-Software bestimmte Eigenschaften auf und erfüllt Kriterien, die denen eines öffentlichen Guts entsprechen. „Öffentliche Güter weisen sich aus durch Nichtrivalität im Konsum (verringert bei zusätzlichem Nutzer die jeden anderen Nutzen zur Verfügung stehende Menge nicht) und durch Nicht-Ausschließbarkeit von der Nutzung (ein zusätzlicher Nutzer kann effektiv nicht von der Nutzung ausgeschlossen werden)“ (Heller und Nuss 2003, S. 394). Wissen, das sich durch Open Source bildet, kann theoretisch nicht verloren gehen, da es für jeden verfügbar und zugänglich gemacht ist, wie es durch die Lizenz vereinbart ist. Open Source akkumuliert auf diese Weise in hohem Maße Wissen und Innovationen. Für diesen Prozess ist eine gut funktionierende Kooperation zwischen den einzelnen Teilnehmern eine der entscheidenen Voraussetzungen. Jeder Einzelne, der die Software weiter entwickelt, leistet seinen Beitrag. Offensichtlich wirken hier andere Anreize, um Wissen, Ideen und Innovationen hervorzubringen.

Die schützenden Instrumente für „geistiges Eigentum“ wirken hier nicht, da Open Source die Eigenschaften eines öffentlichen Guts annimmt und die Nutzungs- und Verwertungsrechte durch die Lizenz klar vereinbart sind. Jeder Einzelne kann durch die Offenlegung des Quelltextes bestehende Software modifizieren, weiterentwickeln und diese nutzen. Da diese Möglichkeit jedem offensteht, kann sich zum einen auf diese Weise sehr unterschiedliche Software entwickeln, zum anderen profitieren die Nutzer davon. Bei Open Source wird in kooperativen Prozessen neues Wissen und Software erzeugt, welches jedem frei zugänglich und zur Verfügung stehen. Dieser Prozess funktioniert jenseits von den Schutzinstrumenten, wie sie bei „geistigem Eigentum“ geregelt sind.

Seit ein paar Jahren zeichnet sich zudem ab, dass das Konzept Open Source nicht nur bei Software sehr gut funktioniert. In ähnlicher Form hat es sich erfolgreich auch auf andere Bereiche kulturellen und gesellschaftlichen Lebens übertragen. So werden im nachfolgenden Kapitel „Open Content“ einige alternative Möglichkeiten

beschrieben, wie Wissen jedem frei zugänglich und verfügbar gemacht werden kann.

4. Open Source – zwischen Geschichte und Zukunft

Open Source besitzt Eigenschaften eines öffentlichen Guts. Jeder kann die Software nutzen, vervielfältigen und weiterverbreiten. Dies fördert den Austausch und Aufbau von Wissen in Form von offenem Quelltext. Mitte der 80er Jahre des letzten Jahrhunderts schuf Richard Stallman die Voraussetzungen mit seiner Initiative freier Software. Ihm ging es um die Freiheit, kooperativ Wissen zu schaffen, dies zu teilen und zu verbreiten. Die Teilnehmer von freier Software sehen in diesen Grundsätzen ein wesentliches Recht: „We are doing that for a reason. The reason [. . .]is to protect the ethical right to share information" (Moglen 2003, S. 2). Dem Ursprung dieser Forderung von freier Software widmet sich Christian Imhorst im ersten Beitrag „Anarchie und Quellcode". So beschreibt er, wie sich durch den Zugang zu den ersten Computern an amerikanischen Universitäten eine Kultur der Hacker etablierte. Diese Programmierer waren überzeugt davon, dass jede Software und Information frei sein sollte. So geht er auf die anarchistischen Ansichten dieser ersten Hacker ein und beschreibt, wie sich diese in Form einer Ethik manifestierten. Einen Schwerpunkt dieser Analyse bildet dabei die Person Richard Stallman, einer der letzten Hacker dieser frühen Bewegung, der durch freie Software die ursprünglichen Ansichten wiederbeleben wollte.

Der nachfolgende Beitrag der Autoren Stefan Meretz und Stefan Merten beleuchtet freie Software aus einer anderen Perspektive. Hier ist der Blick in die Zukunft gerichtet und es wird der kontroversen These nachgegangen, das Konzept von freier Software sei für eine alternative Gesellschaftsform geeignet. Hierzu werden als erstes die Eigenschaften und Prinzipien freier Software näher beschrieben, um dann auf ihre Produktionsweise einzugehen. Diese Überlegungen münden in der These, dass es sich bei freier Software um eine Keimform handelt, also um eine Vorstufe gesellschaftlichen Wandels, die mit der GPL-Gesellschaft als utopisches Modell beschrieben wird.

Der Frage „Open Source – Die Rückkehr der Utopie?" widmen sich die Autoren Christian Görlich und Ludger Humbert im darauffolgenden Artikel. So wird Open Source unter verschiedenen Gesichtspunkten betrachtet und in Bezug zu bestehenden Theorien und Überlegungen klassischer Soziologen und Philosophen eingeordnet. So zeigen die Autoren bei Open Source beispielsweise Elemente eines sozialen Systems auf, welches den Drang nach Kreativität als Basis hat. Weiter werden klassische Normen auf die Open-Source-Bewegung bezogen, die sich zum einen als idealisierte Form einer wissenschaftlichen Gemeinschaft zeigt, zum anderen Züge einer Tauschgesellschaft trägt. Danach widmen sich die Autoren den unterschiedlichen Weltanschauungen von Open-Source-Projekten.

Der wesentliche Erfolg von Open Source ist maßgeblich mit der rasanten Entwicklung des Internets verbunden. Durch die vielfältigen Nutzungsmöglichkeiten, die im Laufe der Zeit entstanden sind, und durch das zugrunde liegende Design des Netzes konnte die Open-Source-Bewegung ihr Potential voll ausschöpfen. Dabei ist das In-

ternet relativ ungeplant entstanden. Wohin und wie es sich entwickeln würde, wussten dessen *Erfinder* nicht. Mit der Frage, wie das Internet in Zukunft gestaltet werden soll, beschäftigt sich der letzte Artikel „Infrastrukturen der Allmende" von Bernd Lutterbeck. Anhand des konkreten Beispiels eines Sees als Allmende werden die zu Grunde liegenden Prinzipien und Konzepte beschrieben, die vielfältigste Nutzungsmöglichkeiten sicherstellen. Mit Hilfe dieses Beispiels und der Betrachtung des Internets als Infrastruktur wird eine Analogie aufgezeigt: In den fundamentalen Prinzipien des Internets finden sich wichtige Voraussetzungen begründet, die deutlich machen, dass ein als Allmende konzipiertes Netz ein hohes Innovationspotential besitzt und zahlreiche Nutzungen zulässt.

Literaturverzeichnis

Euchner, W. (Hrsg.) (1992), *John Locke, Zwei Abhandlungen über die Regierung (Two Treatises of Government 1690)*, Suhrkamp, Frankfurt a. Main.

Grassmuck, V. (2004), *Freie Software: Zwischen Privat- und Gemeineigentum*, Bundeszentrale für Politische Bildung, Bonn.

Heller, L. und Nuss, S. (2003), Open Source im Kapitalismus: Gute Idee – falsches System?, *in* B. Lutterbeck und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, S. 385–405.

Lessig, L. (2001), *The Future of Ideas – The Fate of the Commons in a connected World*, Vintage Books.

Moglen, E. (2003), Freeing the Mind: Free Software and the Death of Proprietary Culture, *in* 'Fourth Annual Technology and Law Conference', University of Maine Law School, Portland.

Vaidhyanathan, S. (2003), 'The state of copyright activism', *First Monday* . http://www.firstmonday.org/issues/issue9_4/siva/index.html [31. Jan 2005].

Anarchie und Quellcode – Was hat die freie Software-Bewegung mit Anarchismus zu tun?

CHRISTIAN IMHORST

(FDL 1.2 siehe Seite 466)

Wie anarchistisch ist die Hackerethik? Warum bezeichnet sich Richard Stallman, eine der herausragendsten Figuren der freien Software-Bewegung, selbst als Anarchist? Wir dürfen uns die Anarchisten der freien Software nicht klischeehaft als Chaoten mit zerzausten Haaren, irrem Blick und Armen voller Bomben vorstellen. Ganz im Gegenteil: Sie fordern eine neue Ordnung des „geistigen Eigentums" im Sinn der Hackerethik – Der Zugriff auf Wissen soll frei, dezentral, antibürokratisch und antiautoritär sein. Genau in diesen Forderungen hat die Hackerethik ihre anarchistischen Momente und ihre Bedingung ist der amerikanische Anarchismus. Während der Anarchismus in Europa längst verschwunden ist, hat er in der amerikanischen Tradition überdauert. Aus dieser Tradition heraus reagierten die ersten Hacker am MIT mit praktischem Anarchismus auf die Autoritäten, die ihren Zugang zum Computer beschränken wollten. Für Stallman war das Hackerparadies am MIT der lebendige Beweis dafür, dass eine anarchistische Gemeinschaft möglich ist. Nach ihrem Vorbild gründete er eine neue Hackergemeinschaft: Das GNU–Projekt. Die freie Software-Bewegung in Form von GNU, BSD oder *Open Source Initiative* ist die radikale, anarchistische Kritik an der heutigen Ordnung des „geistigen Eigentums". Im Gegensatz zu den BSD-Vertretern oder dem marktwirtschaftlichen Anarchismus Eric Raymonds plädiert Stallman für einen genossenschaftlichen Anarchismus, dass wir uns freiwillig zusammensetzen und ausdenken sollen, wie wir durch Zusammenarbeit für alle sorgen können.

1. Einleitung

In einem New Yorker Restaurant saßen zwei Männer beim Mittagessen zusammen und überlegten sich die nächsten Schritte ihrer kleinen Revolution. Einer der beiden, Eben Moglen, dachte kurz darüber nach, wie sie wohl auf die Leute wirken mochten,

die vorbei gingen: „Hier sind wir, zwei kleine bärtige Anarchisten, die sich ihre nächsten Schritte überlegen. Jeder, der unser Gespräch belauscht, wird denken, dass wir verrückt sind. Aber ich weiß: Ich weiß, dass wir hier an diesem Tisch eine Revolution vorbereiten." Und sein Gegenüber, Richard Stallman, sollte diese Revolution möglich machen (Vgl. Williams 2002, S. 184).

Doch nicht nur Eben Moglen, Rechtsprofessor an der Columbia Law School, sieht sich als Anarchist. Richard Stallman, eine der herausragendsten Figuren der freien Software-Bewegung, sieht sich selbst auch so. Wir dürfen uns die Anarchisten der freien Software dabei nicht klischeehaft als Chaoten mit zerzausten Haaren, irrem Blick und Armen voller Bomben vorstellen. Ganz im Gegenteil: Statt Chaos fordert Stallman eine neue Ordnung des „geistigen Eigentums" im Sinne der Hackerethik – der Zugriff auf Wissen soll frei, dezentral, antibürokratisch und antiautoritär sein. Stallman hat als Amerikaner weniger Scheu, sich als Anarchist zu bezeichnen, weil der Anarchie-Begriff in Amerika ein anderer ist als in Europa. Für David DeLeon, Professor für Geschichte an der Howard Universität in Washington D. C., ist der Anarchismus in Amerika in seinem Buch „The American as Anarchist" die einzig radikale konstruktive Kritik an der liberalen Gesellschaft der Vereinigten Staaten. Darüber hinaus, so meine These, ist der amerikanische Anarchismus Bedingung für die Hackerethik (Imhorst 2004).

Stallmans Botschaft ist eine radikal-politische Botschaft, denn es geht um Privateigentum, einen Eckpfeiler der Gesellschaft, in der wir leben. „Geistiges Privateigentum" in Form von Software ist die Gelddruckmaschine des ausgehenden 20. und des beginnenden 21. Jahrhunderts. Schließlich hat der reichste Mann der Welt, Bill Gates, seinen Reichtum weder mit Öl, Gold oder Aktienspekulationen erworben, sondern mit Software. Mit Copyrights und Patenten auf „geistiges Eigentum" in Form von Software kann man seit den 1980er Jahren Milliarden US-Dollar verdienen.

Die Gegner von Stallman werfen ihm vor, dass er das „geistige Eigentum" abschaffen wolle und mit seiner freien Software-Bewegung einer kommunistischen Utopie nachhänge. Er selbst sieht sich nicht als Kommunist oder antikapitalistischer Staatsfeind, der das Eigentum abschaffen will. Stallmans Lizenz, die GNU General Public License (GPL), ein politischer Ausdruck des *free spirit* in der freien Software–Bewegung, spricht auch nicht von der Abschaffung des „geistigen Eigentums". Im Gegenteil, sie will bestimmtes „geistiges Eigentum" schützen.

Der Programmierer freier Software verschenkt mit der GPL die Kontrolle über sein Werk, nicht aber das Werk als solches. Er behält die Autorenschaft über sein Programm. Dem Benutzer der Software wiederum werden bestimmte Freiheiten gewährt, wie die Freiheit, das Werk zu modifizieren und verändert zu veröffentlichen. An diese Freiheit ist nur eine Bedingung geknüpft: Das veränderte Werk muss wieder unter der GPL stehen. Ähnliche Lizenzen gibt es auch für Bücher, Musik und andere Formen „geistigen Eigentums". Diese Freiheiten können von keinem zurückgenommen werden. Freie Software soll nicht Eigentum eines Einzelnen, sondern das Eigentum von allen sein. Ihr Gegenstück ist die proprietäre Software. Ein proprietäres Programm wie Microsoft Word ist Privateigentum der Firma Microsoft. Wer sich Word installiert, hat nur ein Nutzungsrecht an dem Programm. Die umfangreiche Lizenz soll Word

davor schützen, dass das Programm einfach weitergegeben oder modifiziert wird. Die GPL dagegen ermutigt den Anwender zur Modifikation und zur Weitergabe der Software. Niemand ist vom Eigentum an GPL-Software ausgeschlossen. Ihre Verbreitung kann deshalb von niemandem kontrolliert werden. Wer sie haben möchte, kann sie einfach kopieren und weitergeben, wodurch die Verfügbarkeit von GPL-Software sehr schnell wächst. Die GPL verhindert zwar, dass Menschen von dem Gebrauch freier Software ausgeschlossen werden, aber sie schließt auf der anderen Seite ebenfalls aus, dass jemand aus freier Software proprietäre macht. Niemand kann daran gehindert werden, das freie Betriebssystem GNU/Linux zu benutzen, und niemandem kann es weggenommen werden. Jeder, der GNU/Linux aus dem Internet herunter lädt, auf seinen Rechner installiert, Kopien davon verschenkt oder verkauft, dem gehört es auch. In diesem Sinne ist die GPL eher eine Anti-Lizenz, weshalb Stallman von ihr auch lieber als *Copyleft* spricht anstatt von einem *Copyright*.

Grundlegend für Stallmans politische Philosophie ist die Hackerethik. Ein Kodex, den sich eine Gruppe von Computerfreaks am Massachusetts Institute of Technology (MIT) gegen Ende der 1950er Jahre gegeben hatte. Sie lernten gemeinsam, die ersten Computer am MIT zu programmieren und das Wissen darüber miteinander zu teilen. Das gemeinsame Programmieren, Lernen und den freien Austausch von Wissen nannten sie hacken und sich selbst Hacker, bevor Journalisten Computerpiraten so nannten. Die Hackerethik hat ihre anarchistischen Momente in den Forderungen nach Freiheit und Dezentralisierung, sowie in ihren antibürokratischen und antiautoritären Bestrebungen.

Während der Anarchismus in Europa weitestgehend verschwunden ist, hat er in der amerikanischen Tradition überdauert. Dafür macht DeLeon in „The American as Anarchist" historisch drei wesentliche Eigenschaften der amerikanischen Lebensweise verantwortlich: Den radikalen Protestantismus als nach innen gekehrte Religion, das weite Siedlungsgebiet, in denen sich Gemeinschaften der Kontrolle des Staates entziehen konnten und den amerikanischen Anarchokapitalismus:[1]

> „Unsere Radikalen konzentrieren sich auf Emanzipation, sie wollen die Ketten der Herrschaft zerreißen, anstatt neue zu schmieden. Sie sind Befreier und keine Gründer von Institutionen; sie treten für die Rechte von Frauen, Schwulen, Schwarzen und für die Befreiungstheologie ein; sie sind Propheten, keine Priester; Anarchisten und keine Verwalter. Im allgemeinen vermuten sie, dass der befreite Geist wenig bis gar keine Führung braucht." (DeLeon 1978, S. 4)

2. Amerikanischer Radikalismus

Der Anarchismus in den USA ist nach zwei Jahrhunderten Unabhängigkeit fundamental vom europäischen oder russischen Anarchismus zu unterscheiden. Die Einwohner der USA gaben sich eine eigene nationale Identität, als sie sich von Europa emanzipierten. Dabei schufen sie einen eigenen liberalen Radikalismus des „new lands, new

1 Alle Übersetzungen aus englischen Quellen durch den Autor.

men, new thoughts“. Der amerikanische Radikalismus war neu und keine Variation des europäischen Radikalismus. Die amerikanischen Anarchisten wollten niemals alle Autorität abschaffen. Sie waren Vertreter einer neuen Form von Ordnung, die des amerikanischen Anarchismus. Problematisch an dem Begriff des Anarchismus ist, dass er selbst niemals eine Doktrin oder feststehende Lehre sein kann. Der Anarchismus kann von jedem seiner Anhänger neu überdacht und wieder anders vertreten werden. In Amerika führte das zu der ausgeprägten Spaltung in einen „rechten“ und einen „linken“ Anarchismus. Konsistenz in der politischen Theorie darf man von Anarchisten nicht erwarten, denn der Anarchismus ist eher eine sich stets neu vollziehende Situationsanpassung. Doch gerade diese Anpassungsfähigkeit führt dazu, dass uns der Anarchismus durch die Geschichte der Menschheit begleitet und zuletzt verstärkt in der Hippie-Bewegung zum Ausdruck kam.

Die anarchistischen Hippies in Kalifornien waren die Pioniere der politischen Gegenkultur der 1960er Jahre. Sie beeinflussten die linken Bewegungen auf der ganzen Welt. Mit ihrer politischen Form der „Direkten Aktion“ – der ältere Begriff aus der anarchistischen Tradition ist „Propaganda der Tat“ – organisierten sie Kampagnen gegen Militarismus, Rassismus, sexuelle Diskriminierung und so fort. Die englischen Soziologen Richard Barbrook und Andy Cameron bezeichnen sie in ihrem Aufsatz „Kalifornische Ideologie“ als „Liberale im sozialen Sinne des Begriffs.“ Die Hippie–Bewegung schuf keine Hierarchien wie die traditionelle Linke, sie schufen kollektive und demokratische Strukturen.

> „Überdies verband die kalifornische Linke den politischen Kampf mit einer Kulturrebellion. Anders als ihre Eltern weigerten sich die Hippies, nach den strengen gesellschaftlichen Konventionen zu leben, in die organisierte Menschen seitens des Militärs, der Universitäten, der Unternehmen und selbst der linksgerichteten politischen Parteien gezwungen wurden. Statt dessen zeigten sie ihre Ablehnung der ordentlichen Welt durch lässige Kleidung, ihre sexuelle Promiskuität, ihre laute Musik und ihre entspannenden Drogen.“ (Barbrook und Cameron 1997)

3. Anarchistische Hacker

Für sexuelle Promiskuität, laute Musik und entspannende Drogen waren die ersten Computerfreaks am MIT nicht unbedingt zu gewinnen. Trotzdem haben sie einiges mit den Hippies gemeinsam, nämlich die anarchistische Ablehnung von autoritären und bürokratischen Strukturen und die Forderung nach ihrer Überwindung. Nach Steven Levys Buch „Hackers – Heroes of the Computer Revolution“ entwickelte sich die Subkultur der Hacker in den späten 1950er Jahren am MIT. Im Frühling 1959 bot die Universität den ersten Kurs in Computerprogrammierung an. Einen Computer zu bedienen war zu der Zeit mit großem Aufwand verbunden. Befehle gab man in diese riesigen Maschinen noch über Lochkarten ein, Bildschirme hatten sie nicht. Bis man allerdings so weit war, dass man einen Computer mit Lochkarten programmieren durfte, musste man zuerst an den Ingenieuren vorbei, die sich selbst als Priesterschaft

bezeichneten und über den Computer wachten. Wollten die ersten Hacker wie Peter Samson, Bob Saunders oder Alan Kotok an einen dieser Millionen Dollar teuren IBM-Computer, wurden sie sehr schnell von der Priesterschaft vertrieben.

> „An den IBM-Computern arbeiten zu wollen, war frustrierend. Es gab nichts schlimmeres, als die lange Wartezeit zwischen der Eingabe der Lochkarten und der Ausgabe des Ergebnisses. Wenn du nur einen Buchstaben in einer Instruktion falsch gesetzt hast, ist das ganze Programm abgestürzt und du musstest den ganzen Prozess von vorne beginnen. Das ging Hand in Hand mit der erdrückenden Ausbreitung gottverdammter Regeln, die die Atmosphäre des Rechenzentrum immer weiter durchdrangen. Die meisten Regeln wurden extra erfunden, um junge verrückte Computerfans wie Samson und Kotok und Saunders physisch von den Rechnern fernzuhalten. Die strengste Regel war, dass niemand den Computer wirklich berühren oder sich an ihm zu schaffen machen durfte. Das war etwas, was sie wahnsinnig machte." (Levy 1984, S. 27)

Samson und Kotok reichte es nicht, sich die Computer nur anzusehen. Sie wollten wissen, wie sie funktionierten. So belegten sie den Computerkurs, um endlich herauszufinden, wie Computer arbeiten. Die strengen Regeln, die im Bereich der Lochkartencomputer herrschten, und die Priesterschaft, die über die Rechner wachte, machten das Hacken schwer. In Gang kam die Subkultur der Hacker erst mit einer neuen Computergeneration, die keine Lochkarten mehr brauchte. Wollte man an den neuen Computern arbeiten, gab es auch nicht so eine große bürokratische Hürde zu überwinden wie bei den alten IBM-Maschinen. Dadurch, dass man Programme direkt mit Tastatur und Bildschirm am Computer starten konnte, ohne einen Stapel Karten einlesen zu müssen, inspirierten die neuen Computer die Programmierer zu einer neuen Form des Programmierens, und die Hacker waren ihre Pioniere. An einem Computer wurden außerdem mehrere Bildschirme und Tastaturen angeschlossen, so dass mehrere Leute den Rechner nutzen konnten, indem sie sich die Rechenzeit teilten.

4. Hackerethik

In ihrem täglichen Kampf um Rechenzeit und gegen die Autoritäten, die sie daran hinderten, zu programmieren, entwickelten die jungen Hacker ihre eigene Ethik. Noch waren sie wenige, und sie nahmen ihre Hackerethik noch nicht ganz so ernst, wie es später bei der Fall sein wird. Die Hackerethik wurde nicht als Manifest veröffentlicht, sie wurde in den ersten Jahren mündlich überliefert. Sie wurde auch nicht diskutiert, sondern von den Hackern, die sie annahmen, wie Axiome hingenommen. Die wichtigsten Punkte der Hackerethik sind, dass der Zugriff auf Computer unbegrenzt, total und alle Information frei sein soll. Autoritäten soll misstraut und Dezentralisierung gefördert werden (Levy 1984, S. 40 ff.).

Vor allem die Universitätsbürokratie hat es Hackern sehr schwer gemacht, wertvolle Rechenzeit an den wenigen Computern zu bekommen. Offene Systeme ohne Büro-

kratie und Autoritäten ermöglichten es ihnen, viel produktiver an den Computern zu sein. Ohne autoritäre Aufsicht durch die IBM-Priester konnten sie am Computer viel mehr leisten. Sobald sie hinter der Konsole einer IBM-Maschine saßen, hatten sie Macht über sie. So ist es fast natürlich, dass sie jeder anderen Macht misstrauten, die sie ohnmächtig machen und die Hacker davon abhalten wollte, ihre Macht über den Computer voll auszunutzen.

Levy hält in „Hackers" die Geschichte der Hackerkultur und ihrer Hackerethik am MIT bis zu ihrem damals vorläufigen Ende 1984 fest. Ein ganzes Kapitel befasst sich mit Richard Stallman, den Levy als den letzten wahren Hacker sieht. Dort sagt Stallman, dass die Hackerkultur am MIT das lebende Beispiel für eine anarchistische und großartige Einrichtung gewesen sei, bevor sie ausgelöscht wurde (Levy 1984, S. 423). Stallman gründete nach dem Vorbild der Hackerkultur eine neue Gemeinschaft, das GNU-Projekt, einer der wichtigsten Pfeiler der freien Software-Bewegung.

Denn die freie Software-Bewegung ist mehr als das GNU-Projekt. Die wichtigste Trennlinie verläuft wohl zwischen den Anhängern von BSD- und GPL-Lizenzen. Die Berkeley Software Distribution (BSD) ist eine Version des Betriebssystems Unix, die an der Universität von Kalifornien in Berkeley Ende der 1970er Jahre entstanden ist. BSD bezeichnet heute eine ganze Reihe von Unix-Ablegern, wie FreeBSD, NetBSD oder OpenBSD. Im Gegensatz zur GPL erlaubt es die BSD-Lizenz, dass der freie Quellcode unter bestimmten Bedingungen in proprietärer Software verwendet werden darf. Es musste bis vor kurzem lediglich der Universität von Kalifornien gedankt werden. Am Anfang eines Software-Projekts kann es eine große Streitfrage sein, ob es die GPL- oder die BSD-Lizenz haben soll. Entscheidet man sich für die BSD-Lizenz, gibt man anderen die Möglichkeit, kommerzielle Versionen aus der eigenen Arbeit zu entwickeln. Entscheidet man sich für die GPL, ist jeder verpflichtet, Quellcode in das Projekt zurückfließen zu lassen. Der Journalist Peter Wayner schreibt in seinem Buch „Kostenlos und Überlegen! Wie Linux und andere freie Software Microsoft das Fürchten lehren" über diesen Streit:

> „Wer sich der GPL anschließt, hat wahrscheinlich auch weniger Probleme mit Richard Stallman, oder sieht zumindest davon ab, öffentlich über ihn herzuziehen. GPL-Anhänger neigen zur individualistischen Bilderstürmerei, halten die eigenen Projekte für ziemlich kultig und werden von einer merkwürdigen Mischung aus persönlicher Überzeugung und 'Was-bin-ich-doch-für-ein-cooler-Typ'-Hysterie angetrieben. Anhänger von BSD-Lizenzen machen dagegen einen eher pragmatischen, organisierten und konzentrierten Eindruck." (Wayner 2001, S. 159)

Die BSD-Anhänger treiben kaum einen Kult um ihre Lizenz. Meist streichen sie nur die Freiheit ihrer BSD-Lizenz gegenüber der GPL-Lizenz heraus. Sie haben auch keine Coverstars wie Richard Stallman oder Linus Torvalds. Von der Presse werden die BSD-Projekte deshalb auch meist ignoriert. Stallmans Kreuzzug zur Befreiung des Quellcodes können BSD-Anhänger daher nicht viel abgewinnen.

Stallmans Sicht ist radikaler. Er will ein System freier Software, das Unix zwar ähnlich, aber besser sein soll. Deshalb nannte er seine Arbeit GNU, ein rekursives

Akronym für „'GNU's Not Unix". Ziel des GNU-Projekts ist es seitdem, ein vollständig freies und funktionstüchtiges Betriebssystem mit allen notwendigen Programmen zu entwickeln. Es sollte von Anfang an mehr als nur ein Sammelbecken für freie Software sein. GNU ist ein System freier Software, das jede proprietäre Software durch GNU-Software ersetzen soll. Mit der Gründung des GNU-Projekts beginnt Stallmans Kreuzzug. Die Freiheit, die vorher nur in der Hackerethik kodifiziert war, wurde nun in einem Vertrag zwischen dem Autor und dem Nutzer rechtsverbindlich in der GNU General Public License festgehalten. Die GPL soll das System freier Software davor schützen, ausgebeutet zu werden.

5. Der Anarchismus der *Open Source Initiative*

Am 15. Mai 1969 stürmten bewaffnete Polizeieinheiten auf Befehl des Gouverneurs von Kalifornien, Ronald Reagan, den People's Park in der Nähe des Campus der Universität von Kalifornien, um gegen protestierende Hippies vorzugehen. Dabei wurde ein Mensch getötet und über hundert verletzt. Das konservative Establishment mit Gouverneur Reagan und die Gegenkultur der Hippies schienen zwei antagonistische Gegensätze zu sein. David DeLeon findet in seinem Buch „The American as Anarchist" allerdings heraus, dass Gouverneur Reagan und die Hippies eher zwei Extreme desselben amerikanischen Anarchismus sind.[2] Für DeLeon ist der Anarchismus in den USA die einzige radikale Kritik von Rechten und Linken an der liberalen amerikanischen Gesellschaft. Er nennt die beiden Flügel auch „rechte und linke Libertäre (*libertarians*)". Wobei Libertär nur ein anderes Wort für Anarchist ist.

Wendet man die Theorie von DeLeon auf die Anhänger der GPL und die Verfechter der Open-Source-Definition an, dann sind sie ebenfalls zwei Extreme des amerikanischen Anarchismus. Eric S. Raymond nennt diese beiden Extreme in seinem Aufsatz „The Cathedral and the Bazaar" das Kathedralen- und das Basar-Modell (Vgl. Raymond 2001).

Frederick P. Brooks stellte in seinem 1975 erschienen Buch „The Mythical Man–Month: Essays on software engineering" (Vgl. Brooks 1995) ein Gesetz auf, nachdem sich jedes Softwareprojekt verzögert, je mehr Entwickler an dem Projekt beteiligt sind. Wie viele andere Hacker glaubte auch Eric Raymond als ehemaliges Mitglied des GNU-Projekts, dass viele Köche den Brei verderben würden und nach dem Brook'schen Gesetz ein Software-Projekt davon profitiere, wenn nur wenige an ihm beteiligt seien. Auch die Software-Projekte des GNU-Projekts bestehen nach diesem Gesetz aus nur wenigen Entwicklern. Zu Raymonds Erstaunen bewies Linus Torvalds mit der schnellen Veröffentlichung von Linux das genaue Gegenteil: Je mehr Hacker er einlud, im Linux-Projekt mitzumachen, desto besser wurde es.

Raymond schrieb seine Beobachtungen in dem Aufsatz „The Cathedral and the Bazaar" nieder, indem er die unterschiedlichen Stile in der Leitung von GNU-Projekten mit dem Linux-Projekt kontrastierte. Den Aufsatz entwickelte er aus einer Rede, die

2 In seinem Wahlkampf zum Präsidenten trat Reagan mit anarchokapitalistischen Wahlkampf-Slogans entschieden für das freie Unternehmertum und für eine weitere Deregulierung des freien Marktes ein.

er zum ersten Mal 1997 auf einem Linux-Kongress in Deutschland hielt. Der Titel des Aufsatzes stammt von seiner zentralen Analogie: GNU-Programme seien eher wie beeindruckende Kathedralen, zentral geplante Monumente der Hackerethik, gebaut für die Ewigkeit. Das Linux-Projekt gleiche eher einem großen Basar mit schwatzenden Händlern. In dieser Analogie impliziert ist auch ein Vergleich zwischen Stallman und Torvalds. Stallman sei der klassische Vertreter des Architekt einer Kathedrale. Er ist ein Programmier-Guru, der für 18 Monate verschwinden könne, um mit einem genialen C-Compiler wieder aufzutauchen. Torvalds sei mehr wie der Gastgeber einer Dinnerparty. Diskussionen über das Design von Linux überlässt er den einzelnen Projektgruppen. Er schreitet nur ein, wenn es in einer Gruppe dermaßen Streit gibt, dass sie einen Schiedsrichter braucht. Schließlich entscheidet letztendlich Torvalds darüber, was in den Kernel hinein kommt. Seine wichtigste Aufgabe bestehe darin, den Fluss der Ideen aufrechtzuerhalten.

Seine Analyse brachte Raymond den Ruf ein, ein „Evangelist des freien Marktes" zu sein. Er hält von staatlichen Eingriffen in den Markt nicht viel. Der Einzelne soll generell durch Dereglementierung gestärkt werden, was bei ihm auch Waffenbesitz miteinschließt. DeLeon würde ihn in „The American as Anarchist" eben seine Kategorie der „rechten Libertäre" einsortieren. „Rechte Libertäre" sind Anarchisten, die meinen, dass die Regierung sie in Ruhe lassen solle, so dass sie mit ihrem Geld und ihren Waffen anfangen können, was sie wollen. Als Raymond nun die freie Software-Bewegung und ihren produktiven Anarchismus untersuchte, entdeckte er, was er als rechter Libertarier entdecken wollte: einen unreglementierten freien Markt. Die Grundlage des Erfolges der freien Software-Bewegung ist demnach die Freiheit der Nutzer. Das Basar-Modell spricht somit schon für eine größtmögliche Deregulierung und Freiheit, da viele verschiedene Händler miteinander konkurrieren. GNU-Projekte, oder auch firmeneigene Entwicklungen proprietärer Software, sollen dagegen ähnlich strukturiert sein wie die mittelalterliche Gemeinde: Der Kathedralenbau wird mit dem Geld der Stadt von einer Gruppe von Priestern vorangetrieben, um die Ideen eines Architekten zu verwirklichen. So kann der Kathedralenbau nur gelingen, wenn genügend Geld, ein talentierter Architekt und Arbeiter vorhanden sind. Die vielen verschiedenen Händler auf dem Basar versuchen dagegen, sich gegenseitig aus dem Feld zu schlagen. Der Beste von ihnen hat auch die meisten Kunden, ganz im sozialdarwinistischen Sinn: Der am besten Angepasste überlebt.

Das Problem an Raymonds Aufsatz ist allerdings, dass weder GNU-Projekte als reine Kathedralen noch das Linux-Projekt wirklich als Basar erscheinen. Vor allem dem Linux-Projekt mit seinen Veröffentlichungen in möglichst kurzen Zeiträumen und seinen Tausenden von Mitarbeitern, steht ganz oben voran Linus Torvalds, der entscheidet, was in den neuen Kernel Einzug findet und was nicht. Das Linux-Projekt ist mehr eine Mischform als ein reiner Basar oder eine reine Kathedrale.

In seiner Rede sprach Raymond 1997 noch von freier Software. Ab Februar 1998 ersetzte er in seinem Aufsatz „freie Software" durch „Open Source". Für ihn und einige andere Anhänger der freien Software-Bewegung wurde Stallman immer mehr zum Ärgernis. Sie fanden, dass Stallman für Geschäftsleute wegen seiner politischen Aussagen zu sehr nach Kommunismus roch. Außerdem wollten sie, dass die Bewegung

sich nicht zu sehr auf die GPL konzentrierte. Sie wollten ein System von Software, in das GPL-Software genauso passt wie Software, die unter der BSD- oder ähnlichen Lizenzen fällt und nannten das Ganze Open Source. Volker Grassmuck sagt in „Freie Software – Zwischen Privat- und Gemeineigentum" dazu:

> „'Free' ist nicht nur zweideutig ('Freibier' und 'Freie Rede'), sondern offensichtlich war es in *The Land of the Free* zu einem unanständigen, 'konfrontationellen', irgendwie kommunistisch klingenden *four-letter word* geworden." (Grassmuck 2002, S. 230)

6. Der letzte wahre Hacker

Levy interviewte für sein Buch „Hackers" auch Richard Stallman. Seiner Geschichte widmet er ein ganzes Kapitel, was nicht umsonst mit „Epilogue: The Last Of The True Hackers" überschrieben ist. Denn 1984 stand es schlecht um die freie Software. Stallman gehört zur Generation der ersten Hacker, die riesige IBM-Maschinen an amerikanischen Universitäten programmiert hatten. Die jungen Leute, die in den 1980er Jahren in den Computerräumen der Universitäten auftauchten, lernten an ihren Heimcomputern unberührt von jeder Hackerethik und Hacker-Gemeinschaft das Programmieren.

> „Diese neuen Leute schrieben wie ihre Vorgänger im Rechenzentrum aufregende Programme. Doch mit ihnen hielt auch etwas neues Einzug – als ihre Programme auf den Computermonitoren auftauchten, waren sie mit Copyrights versehen. Für Stallman, der daran festhielt, dass alle Information frei fließen sollte, war das Blasphemie. 'Ich denke nicht, das Software jemanden gehören sollte', da diese Praxis die Menschlichkeit als Ganzes sabotiert. Es hindert Leute daran, den kompletten Nutzen aus der Software zu ziehen." (Levy 1984, S. 419)

Diese neuen Hacker interessierten sich auch nicht sonderlich für die Hackerethik. Stallman hatte im Rechenzentrum des MIT gelernt, dass eine anarchistische Institution möglich ist. Nur fehlten ihm die Mitstreiter aufgrund der Dezentralisierung der Hacker durch die Heimcomputer. Anfang der 1980er Jahre fühlte er sich als letzter Anhänger einer scheinbar toten Bewegung, die sich an den anarchistischen Grundsätzen der Hackerethik ausrichtete. Diese Bewegung wollte er wiederbeleben. Mit der freien Software-Bewegung wird die Hackerkultur wiedergeboren und Stallman tritt an, den Quellcode von proprietären Lizenzen zu befreien.

Die freie Software-Bewegung in Form von GNU, BSD oder *Open Source Initiative* ist die radikale, anarchistische Kritik an der heutigen Ordnung des „geistigen Eigentums" nicht nur in der liberalen Gesellschaft der Vereinigten Staaten, sondern an dessen Ordnung in der ganzen globalisierten Welt. Im Gegensatz zu den BSD-Vertretern oder dem marktwirtschaftlichen Anarchismus Eric Raymonds von der *Open Source Initiative* plädiert Stallman für einen genossenschaftlichen Anarchismus, der

zumindest in Bezug auf das „geistige Eigentum“ frei nach dem französischen Anarchisten Jean-Pierre Proudhon sagt, dass Eigentum Diebstahl sei. Für viele ist die Forderung nach Abschaffung des geistigen Privateigentums heute undenkbar. Dabei war noch vor einem halben Jahrtausend die Einführung von Eigentum undenkbar. So sagt Jeremy Rifkin in „Access – Das Verschwinden des Eigentums“:

> „Dass wir Marktsystem und Warentausch hinter uns lassen, [. . .] ist derzeit für viele Menschen noch genauso unvorstellbar, wie es die Einhegung und Privatisierung von Land und Arbeit und damit ihre Einbindung in Verhältnisse des Privateigentums vor einem halben Jahrtausend gewesen sein mögen.“ (Rifkin 2000, S. 24)

Stallman und die GNU-Fraktion der freien Software-Bewegung wollen „geistiges Eigentum“ nicht nur in Gestalt von Software, sondern auch in der Gestalt von Büchern oder Musik von proprietären Lizenzen befreien. Warum, das sagt Stallman im Gespräch mit Spiegel Online: „Ich tendiere mehr zu der linken anarchistischen Idee, dass wir uns freiwillig zusammensetzen und ausdenken sollen, wie wir durch Zusammenarbeit für alle sorgen können“ (Klagges 1996).

Literaturverzeichnis

Barbrook, R. und Cameron, A. (1997), ‘Die kalifornische Idiologie’, Telepolis, http://www.telepolis.de/deutsch/inhalte/te/1007/1.html/ [30. Nov 2004].

Brooks, F. P. (1995), *The Mystical Man-Month: Essays on software engineering*, Addison-Wesley, New York.

DeLeon, D. (1978), *The American as Anarchist*, John Hopkins University Press, Baltimore, MA.

Grassmuck, V. (2002), *Freie Software – Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn.

Imhorst, C. (2004), *Die Anarchie der Hacker – Richard Stallman und die Freie-Software-Bewegung*, Tectum Wissenschaftsverlag, Marburg.

Klagges, H. (1996), ‘Es reicht mir nicht, nur einfach neugierig auf die Zukunft zu sein, ich will etwas ändern. – Interview mit Richard Stallman’, Klagges.com, http://www.klagges.com/pdf/interview_stallman.pdf [30. Nov 2004].

Levy, S. (1984), *Hackers: Heroes of the Computer Revolution*, Anchor Press/Doubleday, New York.

Raymond, E. S. (2001), *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, 1. Aufl., O’Reilly, Sebastopol, CA.

Rifkin, J. (2000), *Access – Das Verschwinden des Eigentums*, Campus Sachbuch, Frankfurt am Main.

Wayner, P. (2001), *Kostenlos und Überlegen! Wie Linux und andere freie Software Microsoft das Fürchten lehren*, DVA, Stuttgart/München.

Williams, S. (2002), *Free as in Freedom – Richard Stallman’s Crusade for Free Software*, O’Reilly, Sebastopol, CA.

Freie Software und Freie Gesellschaft

Stefan Merten und Stefan Meretz

Der weltweite Erfolg freier Software hat die Frage aufgeworfen, ob ihre Prinzipien verallgemeinerbar und auf die gesamte Gesellschaft übertragbar sind. 1999 gründete sich das Oekonux-Projekt rund um diese Frage. Oekonux steht für „Oekonomie und GNU/Linux". Zahlreiche Analysen aus unterschiedlichen Richtungen haben zu Thesen geführt, die viele für bahnbrechend halten und an denen sich viele reiben – innerhalb und außerhalb des Projektes. Der erste Teil des Beitrages beschäftigt sich mit den Eigenschaften freier Software und dessen Prinzipien. So wird zuerst die Produktionsweise von freier Software betrachtet, der Unterschied zwischen einfach und doppelt freier Software gezeigt und anschließend erklärt, warum es sich bei freier Software um eine neue Qualität der Produktivkraftentwicklung handelt. Diese Betrachtungen sollen im letzten Teil dazu dienen, in freier Software eine Keimform für eine neue Gesellschaft zu sehen. Diese Betrachtung wird mit dem Bild der „GPL-Gesellschaft" abgeschlossen.

1. Einleitung

Freie Software hinsichtlich eines möglichen Ausgangspunkts einer neuen, fundamental veränderten Gesellschaftsformation zu untersuchen, hat sich das Projekt Oekonux zur Aufgabe gemacht. Das ganz überwiegend virtuelle Projekt gruppiert sich um mehrere Mailing-Listen und einige Websites. Im nicht-virtuellen Raum trat das Projekt mit bisher drei internationalen Konferenzen an die Öffentlichkeit.[1] Der vorliegende Artikel, der diese zentralen Thesen beleuchtet, wurde in einem offenen Prozess entwickelt,[2] der auch jetzt noch für Beiträge offen ist. Der Artikel steht nicht nur zum Lesen zur Verfügung, sondern kann auch direkt online im Browser Absatz für Absatz kommentiert werden.

1 Informationen zum Projekt und den bisherigen Konferenzen befinden sich unter http://www.oekonux.de/projekt/index.html.

2 Der Beitrag befindet sich online unter unter http://www.opentheory.org/ox_osjahrbuch_2005/.

2. Produktionsweise freier Software

Die Produktionsweise freier Software unterscheidet sich grundsätzlich von der proprietärer Software. Dies betrifft weniger die technischen Verfahren, sondern vor allem die individuelle Motivation und die soziale Organisation. Diese Produktionsweise ist gekennzeichnet durch *Wertfreiheit*, *Selbstorganisation*, *Globalität* und *Selbstentfaltung*. Diese vier zentralen Begriffe sollen einleitend kurz umrissen und im restlichen Text aus verschiedenen Perspektiven weiter erläutert werden (Vgl. Merten 2000, Meretz 2000*b*, 2004).

Die Entwicklung von Software ist mit Aufwand verbunden. Bei freier Software wird dieser Aufwand in der Regel jedoch nicht (monetär) entlohnt. Wie auf vielen anderen Gebieten menschlichen Lebens strengen sich die Menschen hier aus anderen Gründen an, als Geld dafür zu erhalten. Das Resultat dieser Tätigkeit ist deswegen ökonomisch *wertfrei* und unterscheidet sich damit wesentlich von der wertbasierten Arbeit, die auf die Erreichung von Lohn oder Profit abzielt (Vgl. Meretz 2000*a*).

Die sozialen Organisationsformen freier Software sind so verschieden wie die Projekte selbst. Niemand gibt von außen vor, wie etwas zu sein hat. Jedes Projekt organisiert sich selbst und findet die Form, die ihm gemäß ist, oft einfach durch Ausprobieren.

Die *globale Vernetzung* istdas Resultat der Möglichkeiten des Internets. Jedes noch so kleine Projekt, das sich einer der vielen frei zugänglichen Projekt-Infrastrukturen bedient (SourceForge, Savannah etc.) oder selbst eine betreibt, ist weltweit verfügbar. Menschen, die sich noch nie gesehen haben und vielleicht auch niemals sehen werden, können so zusammen etwas Nützliches erschaffen. Ohne das Internet mit seinen vielfältigen Diensten wie E-Mail, FTP oder WWW wäre freie Software in ihrer heutigen, entfalteten Form nicht denkbar.

3. Hauptantrieb Selbstentfaltung

Selbstentfaltung ist ein zentraler Begriff für das Verständnis freier Software. Selbstentfaltung meint nicht einfach „Spaß haben“ und besitzt eine individuelle und eine gesellschaftliche Dimension. Individuell meint Selbstentfaltung das persönliche Entfalten der eigenen Möglichkeiten, mithin das Entwickeln der eigenen Persönlichkeit. Eine so verstandene Entfaltung der Persönlichkeit hat verschiedene Formen der Entäußerung: produktive, reproduktive, technische, kulturelle, kommunikative, konsumtive etc. Diese können für andere nützlich sein (Vgl. Himanen 2001).

Die gesellschaftliche Dimension der Selbstentfaltung betrifft die Abhängigkeit der eigenen Entfaltung von der Entfaltung der anderen. Ich kann mich nur entfalten, wenn die anderen es auch tun. Die anderen – potenziell alle anderen – sind meine Entfaltungsbedingung, wie ich umgekehrt Entfaltungsbedingung für die anderen bin. Es entsteht eine positive Rückkopplung: Mein Bestreben richtet sich darauf, dass die anderen sich entfalten können, damit ich mich entfalten kann. Würde ich mich nur darauf konzentrieren, was ich zu tun wünsche und die anderen ignorieren oder gar ausgrenzen, dann würde ich mir selbst schaden.

Diese Dynamik können wir, mehr oder weniger ausgeprägt, bei freier Software beobachten. Die positive Rückkopplung kommt zustande, weil und wenn es keine dritten, entfremdeten Gründe gibt, tätig zu werden. Da freie Software nicht für den Verkauf produziert wird, gibt es keine entfremdeten Gründe, sondern nur jeweils meine Gründe, freie Software zu entwickeln oder zu unterstützen. Proprietäre Software hingegen wird für einen dritten, fremden, der Software äußerlichen Zweck entwickelt.[3]

Im Projekt Oekonux wurde diese Analyse zu dem Satz verdichtet: „Die Selbstentfaltung des Einzelnen ist die Bedingung für die Entfaltung Aller – und umgekehrt". Besonders deutlich wird die Unterscheidung, wenn man den gleichen Satz für die entfremdete Warenproduktion formuliert. Dort gilt: Die Entwicklung des Einzelnen ist möglich auf Kosten der Entwicklung der anderen – und umgekehrt.

Selbstentfaltung darf nicht mit Selbstverwirklichung verwechselt werden. Während Selbstentfaltung die anderen als Bedingung für die eigene Verwirklichung versteht, blendet Selbstverwirklichung die gesellschaftliche Dimension aus. Selbstverwirklichung ist statisch und begrenzt, sie geht von einer Anlage aus, die verwirklicht werden will und endet mit der Verwirklichung. Selbstentfaltung hingegen ist dynamisch. Jede erreichte Entfaltung ist wiederum nur Bedingung und Möglichkeit neuer Formen der Entfaltung. Eine Gesellschaft der Selbstentfaltung wäre eine reiche Gesellschaft (Gorz 2004).

4. Einfach und doppelt freie Software

In der freien Software-Bewegung wird von verschiedenen Seiten immer wieder betont, dass freie Software Bestandteil von Geschäftsmodellen sein kann. Es soll möglich sein, mit freier Software Geld zu verdienen.

Nun ist klar, dass Geschäftsmodelle, die mit einer Verknappung eines fertigen Produkts operieren, bei freier Software nicht funktionieren können. Verknappung von Informationsgütern ist unter den Bedingungen der weltweiten digitalen Kopie allgemein nur zu erreichen, wenn den NutzerInnen das Recht genommen wird, selbst das Informationsgut weiterzugeben. Die Grundrechte freier Software – uneingeschränkte Einsatzmöglichkeiten, Einsicht in und Möglichkeit zur Anpassung der Quellen, unbeschränkte Weitergabe originaler oder veränderter Versionen – erlauben diese Verknappung jedoch nicht.

Es ist zwar nicht möglich, fertige Produkte direkt zu verkaufen, dennoch gibt es verschiedene Dienstleistungen rund um freie Software, die verkauft werden können: beispielsweise Wartung, Installation oder Zusammenstellen von Distributionen. Diese Geschäftsmodelle arbeiten zwar mit vorhandener, fertiger freier Software, verkauft wird aber letztlich die Dienstleistung. Kaufe ich beispielsweise eine Distribution von SuSE/Novell, so kaufe ich ein Handbuch, CDs und DVDs sowie das Recht auf telefonischen Support für die Installation, das sogar durch einen speziellen, individuellen Code abgesichert ist. Die freie Software, die auf den mitgelieferten Medien

3 Der Markt entscheidet, ob die Software überlebt und wenn sich zu wenige KäuferInnen finden, verschwindet die Software. Das gibt es bei freier Software nicht. Solange sich eine EntwicklerIn für die Software interessiert, gibt es sie – auch, wenn es keine aktuellen NutzerInnen mehr gibt.

enthalten ist, kann ich mir jedoch ganz legal und ohne Bezahlung auch direkt von den SuSE-FTP-Servern downloaden. Tatsächlich bezahlt wird SuSE also für die materiellen Produkte, die sich auf die verteilte freie Software beziehen sowie für Support-Dienstleistung. Grundlage für dieses Geschäftsmodell ist die Dienstleistung der Zusammenstellung und Pflege der Distribution aus dem riesigen Pool vorhandener freier Software. Auch die Anbieter verschiedener Merchandising-Produkte rund um einzelne freie Software-Projekte oder auch Bücher zu freier Software verkaufen nicht freie Software, sondern eben Plüschpinguine.

Ist mit fertiger freier Software selbst kein Geschäftsmodell zu begründen, so gilt das nicht für freie Software, die noch nicht existiert. Es sind durchaus Geschäftsbeziehungen möglich, bei denen freie Software *im Auftrag* erstellt wird. Solche Auftragsarbeiten unterscheiden sich im Falle von Projekten ohne Anbindung zu einer Community von proprietärer Software lediglich durch die Lizenz, unter der das fertige Produkt später steht. Ebenfalls in diese Kategorie fällt die Weiterentwicklung freier Software, die in Firmen für eigene Zwecke durchgeführt wird.

Gegenüber der klassischen freien Software, wie sie ein Richard Stallman oder ein Linus Torvalds entwickelten, gibt es bei den genannten Geschäftsmodellen jedoch einen wichtigen Unterschied. Wie, wohin und wie schnell sich freie Software entwickelt, die ohne externen Auftrag entsteht, liegt allein in der Entscheidung des jeweiligen Projekts. Zu den Freiheiten, die die Lizenzen den NutzerInnen gewähren, tritt in diesen Fällen die Freiheit der EntwicklerInnen, die nicht an Weisungen von Auftraggebern gebunden sind. In Fällen, wo allein die Selbstentfaltung der EntwicklerInnen den Fortgang des Projekts bestimmt, sprechen wir von *doppelt freier Software*.

Demgegenüber sprechen wir von *einfach freier Software*, wenn die EntwicklerInnen in ihren Entscheidungen nicht frei, sondern an einen Auftraggeber gebunden sind. Die EntwicklerInnen entfremden sich in solchen Projekten von ihrem Produkt, da sie auf Grund der Abhängigkeit von der Bezahlung Entscheidungen des Auftraggebers berücksichtigen müssen, die aus ihrer Sicht für das Produkt schädlich sein können. Alle, die schon einmal Software im Auftrag einer Firma hergestellt haben, kennen unzählige Beispiele für von Marketing oder Vertrieb bestimmte Terminpläne, technisch überflüssige Hochglanzfeatures usw. Hier zeigt sich deutlich die Entfremdung vom Produkt, die mit der Erfüllung des dem Produktnutzen äußerlichen Zwecks der Verwertung entsteht.

Betrachten wir freie Software als neue Produktionsweise, so tritt gerade die freie Entscheidung der EntwicklerInnen in den Vordergrund. Nicht getrieben von Marktvorgaben, mithin frei von den Zwängen der Verwertung, können sich die EntwicklerInnen auf die bestmögliche Qualität der Software konzentrieren. Unter anderem ist es möglich, wie jüngst bei GIMP geschehen,[4] sich eine zweijährige Auszeit zu nehmen, in

4 Aus c't 13/2004, S. 80: „Vor-Meilenstein – Open-Source-Bildbearbeitung GIMP: Fit für die Profi-Liga?" Doch die eigentliche Dreijahresleistung der Entwickler fand hinter den Kulissen statt, wie GIMP-Maintainer Michael „Mitch" Natterer gegenüber c't erklärte: „Mit dieser Version wurde die Programmlogik strikt vom Frontend getrennt, kein Quellcode-Stein blieb auf dem anderen. In GIMP 1.2 befanden sich sämtliche Quellcode-Dateien in einem Verzeichnis; niemand wusste genau, was wozu gehört. Kleine Änderungen an einer Stelle konnten anderswo verheerende Auswirkungen haben." Für die Entwickler

der die historisch gewachsene Code-Basis durch ein neues, qualitativ hochwertigeres Fundament ersetzt wird. Anstatt neue Features zu implementieren, wird hier die langfristige Qualität in einer Weise gesichert, wie man sie sich von bekannten proprietären Software-Produkten wünschen würde.

Der qualitative Vorsprung doppelt freier Software ist struktureller Natur und kann auf Grund der in einfach freier Software angelegten Entfremdung von dieser nicht eingeholt werden. Dieser qualitative Vorsprung ist es aber letztlich, der dem Produktivkraftmodell doppelt freier Software den entscheidenden Vorteil vor dem einfach freier oder proprietärer Software gibt. Gäbe es diesen Vorsprung nicht, so hätte freie Software bei Software-NutzerInnen keine Chance gegen proprietäre Software gehabt. Der heute zu beobachtende Erfolg freier Software basiert im Fundament nämlich auf der doppelt freien Software, die teilweise schon vor vielen Jahren geschrieben wurde.

Dennoch treibt auch einfach freie Software insgesamt die Entwicklung in Richtung einer freien Verfügung über freie Produkte voran und in der Praxis mischen sich beide Formen oft. Dies geschieht insbesondere dann, wenn eine doppelt freie Community zu EntwicklerInnen einfach freier Software hinzutritt.

5. Freie Software ist keine Ware

Wollen wir ermessen, inwiefern sich freie Software vom vorherrschenden Wirtschaftsmodell der Marktwirtschaft unterscheidet, so ist es sinnvoll, freie Software mit einem der zentralen Elemente der Marktwirtschaft zu vergleichen: *Der Ware*. Unter Waren verstehen wir in diesem Kontext Güter, die primär zum Zwecke des Verkaufs auf einem Markt produziert werden und sich also von Gütern unterscheiden, die primär aus anderen Gründen produziert werden, z. B. weil sie nützlich sind. Für am Markterfolg orientierte Produkte genügt bei den ProduzentInnen ein *relativer Qualitätsanspruch*, da es lediglich darum geht, in den Augen der avisierten KäuferInnen besser zu sein als die Konkurrenz. Während sich die Qualität von marktorientierten Produkten in dieser Relativität erschöpft und in Monopolsituationen zu deutlich sichtbarer mangelnder Qualität führt, liegt bei einer Produktion, bei der die Nützlichkeit eines Produkts im Vordergrund steht, ein *absoluter Qualitätsanspruch* in der Logik der ganzen Produktionsweise.

Für die Wareneigenschaft ist der Preis der Ware im Übrigen nicht relevant und preislose Waren sind uns spätestens seit den Lockangeboten für Handy-Verträge vertraut.[5]

---

stand außer Frage, dass sie mit dieser Struktur an einem toten Punkt angelangt waren. „Um an wirklich neue Features wie CYMK, 16 Bit oder Ebenengruppen überhaupt denken zu können, musste das Ganze erst einmal auf ein ordentliches Fundament gesetzt werden", konkretisiert Natterer die Probleme des alten GIMP. Heute befinden sich die internen Funktionen von GIMP in 12 klar voneinander getrennten Modulen. Die Oberfläche wurde fast komplett neu geschrieben und ebenfalls in Module aufgeteilt. „Dabei hat fast jede Zeile Quellcode mehr als einmal ihren Platz gewechselt", umreißt Natterer das Mammutprojekt.

5 Ein Sonderfall ist Freeware, die keine freie Software ist. Freie Software zeichnet sich durch freie Verfügung, freie Quellen, freie Änderbarkeit und freie Verteilbarkeit aus. Freeware zeichnet sich dagegen nur durch Kostenfreiheit aus und muss die Punkte, wie sie bei freie Software gefordert sind, nicht erfüllen.

Freie Software hat jedoch in der Regel keinen Preis mehr, sobald sie einmal veröffentlicht ist. Viele, die an die geldbasierte Gesellschaft gewöhnt sind, sind zunächst einmal skeptisch gegenüber dieser Tauschfreiheit. Sie erwarten, dass Güter, für deren Erhalt sie nichts oder unverhältnismäßig wenig geben müssen, entweder Teil der Werbung sind oder sonst einen Pferdefuß haben. Freie Software ist aber weder Werbung noch hat sie sonst einen Pferdefuß. Insbesondere doppelt freie Software ist vielmehr von Anfang bis Ende jenseits des Tauschprinzips angesiedelt. Auch wenn die Teilnahme an einem freien Software-Projekt Geben und Nehmen beinhaltet, so ist der Erhalt von Leistungen jedoch nicht an die Erbringung von Leistungen gekoppelt. Tatsächlich werden die allermeisten NutzerInnen freier Software wenig oder gar nichts zu deren Weiterentwicklung leisten, können sie aber dennoch völlig uneingeschränkt nutzen.

Auf Grund der Konkurrenz sind Betriebsgeheimnisse in der Warenproduktion unerlässlich. Bei freier Software liegen dagegen die Quellen offen vor, sodass es gar keine Geheimnisse geben kann. Alle Interessierten können jederzeit das gesamte Know-how verwenden, das in einer Software enthalten ist.

Gleichzeitig lädt die Offenheit die NutzerInnen ein, die Software zu benutzen und Fehler und Wünsche zu melden, und sie lädt EntwicklerInnen ein, Verbesserungen und Erweiterungen einzubringen. Jeder auch noch so kleine Beitrag bringt alle voran. Freie Software saugt Kreativität und Wissen an. So herrscht Überfluss nicht nur beim Nehmen, sondern auch die Hineingabe ist potenziell unbegrenzt. Freie Software lädt zur *Kooperation* ein, sie funktioniert nach einem *Inklusionsmodell*.

Konkurrenz, also Durchsetzung auf Kosten anderer, wie wir sie zwischen Warenproduzenten erleben, gibt es insbesondere bei doppelt freier Software nicht. Wo es für eine bestimmte Problemstellung mehrere Programme gibt, so beziehen sie sich nicht konkurrenzförmig, also negativ, aufeinander. Entweder existieren die Projekte ohne besondere Beziehung nebeneinander oder es gibt eine mehr oder weniger starke Kooperation zwischen den Projekten. Proprietäre Software muss dagegen nicht nur das Nehmen begrenzen, sondern auch die Hineingabe ist beschränkt, denn nur ausgewählte EntwicklerInnen dürfen in den Quelltext sehen. Unsichere Software ist oft die Folge. Proprietäre Software basiert auf einem *Exklusionsmodell*.

Nun hat es in der Vergangenheit immer wieder Produktionsformen gegeben, die nicht vom Warenmarkt ausgegangen sind. Nicht selten sind Produkte zunächst im Hobbybereich ersonnen worden, und die Wirtschaft hat diese Erfindungen aufgegriffen. In solchen Fällen ist dem Hobbybereich bestenfalls eine Nische geblieben.

Anders bei freier Software. Wurde Software in der Frühzeit der Computer nicht als eigenständige Ware begriffen, so hatte sich in den 80er Jahren des vergangenen Jahrhunderts ein Warenmarkt für Software etabliert. Freie Software, die auf einem anderen Produktivkraftmodell als dem der Warenproduktion basiert, trat nun aus Sicht der NutzerInnen in direkte Konkurrenz zur proprietären Software. Im Gegensatz zu allen früheren Beispielen konnte sich freie Software aber nicht nur eine Nische sichern, sondern wächst im Gegenteil immer weiter und wird nach und nach zu einer ernsten Bedrohung für die proprietäre Software. Das neue Produktivkraftmodell, das wir in freier Software erkennen können, hat das Zeug dazu, die etablierte Warenwirtschaft zu ersetzen.

Freie Software ist nicht zuletzt im *Überfluss* vorhanden. Allein diese Eigenschaft ist ein nachhaltiges Hindernis, freie Software zu einer Ware zu machen. Um den Zusammenhang zwischen Produktion, Konsumption und Gesellschaft übergreifender zu verstehen, lohnt sich eine differenziertere Betrachtung der Begriffe Vorkommen, Begrenztheit und Knappheit.

Mit *Vorkommen* meinen wir, dass ein Gut vorkommt, unabhängig davon, ob wir es brauchen oder nicht. Die gesellschaftliche Dimension ist in diesem Begriff also nicht enthalten. Vorkommen kennt ein absolutes Maß, das z. B. im Begriff des Rohstoffvorkommens gefasst ist. Verleiht man dem Begriff ein zeitliches Maß, so ist er auch auf hergestellte Güter, also Produkte, übertragbar und er bezeichnet die zu einem bestimmten Zeitpunkt existierenden Produkte.

Mit *Begrenztheit* bezeichnen wir das Verhältnis zwischen der Verfügbarkeit eines Gutes und den Bedürfnissen der Menschen, dieses zu erhalten und zu nutzen. Gemessen am Bedarf, kann ein Gut in zu geringer, eben begrenzter Menge zur Verfügung stehen. Solche Begrenzungen können durch gesellschaftliches Handeln abgestellt werden, indem im einfachsten Fall vom begrenzten Gut mehr hergestellt wird. Produktion im allgemeinen Sinne bedeutet immer, gesellschaftlich mit Begrenzungen umzugehen.

Eine besondere Form des Umgangs mit Begrenzungen ist die Warenproduktion. Eine Ware darf nicht frei verfügbar sein, sonst ist sie keine, sie muss knapp sein. *Knappheit* ist eine geschaffene, soziale Form der Warenproduktion. Sie ignoriert wirkliche Begrenzungen und Vorkommen, um daraus die real wirksame Form Knappheit zu machen. Die soziale Form Knappheit produziert die Paradoxie des Mangels im Überfluss. Da abgelöst vom wirklichen Vorkommen, kann sie auch nicht nachhaltig sein.

6. Das Maintainer-Modell

Neben der Wertfreiheit spielt auch die Organisationsform eine große Rolle. Wer länger in der Softwareentwicklung tätig ist, weiß, dass bei Software-Projekten der soziale Prozess bezüglich der Organisation eine wesentliche Rolle spielt. Ist der soziale Prozess schlecht oder gar nicht organisiert, versinkt das interessanteste Projekt im Chaos, und Kreativität und Produktivität der Aktiven werden nachhaltig gestört. Wie ist dieser Prozess in freien Software-Projekten – genauer: in Projekten doppelt freier Software – organisiert?

Der entscheidende Unterschied zwischen der Entwicklung doppelt freier Software und anderer Software-Entwicklung besteht darin, dass alle ProjektteilnehmerInnen ausschließlich auf der Basis von *Freiwilligkeit* am Projekt teilnehmen. Da die Aktiven durch keinerlei entfremdete Anreize, wie zum Beispiel Entlohnung, an das Projekt gebunden sind, können sie das Projekt auch genauso freiwillig wieder verlassen.

So wie die Teilnahme nicht durch äußerliche Aspekte des Projektes bestimmt ist, ist es auch das Projekt insgesamt nicht. Vielmehr kann und muss sich jedes Projekt selbst Ziele setzen, sich *selbst organisieren*. Die Ziele beziehen sich dabei ausschließlich auf das gemeinsame Produkt und dessen Qualität.

Die Bedingungen der Freiwilligkeit der TeilnehmerInnen und der Selbstorganisation des Projekts bilden damit den Rahmen, in dem sich jede Organisation eines doppelt freien Software-Projekts abspielen muss. Wie dieser Rahmen in der Praxis gefüllt wird, ist nicht festgelegt. In sehr vielen Projekten gibt es jedoch das Maintainer-Modell.

Das Maintainer-Modell unterscheidet im Wesentlichen zwei Rollen, den oder die MaintainerIn und andere TeilnehmerInnen. Die Aufgaben der MaintainerIn bestehen im Wesentlichen darin, das Projekt generell auf Kurs zu halten. Die MaintainerIn entscheidet verbindlich über die grundsätzliche Richtung, in die die Software des Projekts weiterentwickelt werden soll, kümmert sich um die Einhaltung projektinterner Standards und darum, dass das Projekt sich bei Bedarf überhaupt weiterentwickelt. Nicht selten regelt die MaintainerIn auch die Außenkontakte für das Projekt. MaintainerInnen kommandieren jedoch nicht die anderen TeilnehmerInnen, vielmehr leisten diese freiwillig Beiträge zum Projekt in Form von Code, Dokumentation, Bug-Reports und vielem anderen mehr (Vgl. Raymond 2000).

Die speziellen Rahmenbedingungen doppelt freier Software führen zu Strukturen, die sich wesentlich von den bekannten Leitungsformen bei herkömmlicher Software–Entwicklung unterscheiden. Da die TeilnehmerInnen freiwillig am Projekt teilnehmen, können Entscheidungen nur getroffen werden, wenn der *Konsens* der wichtigen TeilnehmerInnen erreicht wird. Konsens meint hier nicht, dass alle zustimmen müssen (Einstimmigkeit), sondern Konsens ist vielmehr erreicht, wenn die TeilnehmerInnen einer Entscheidung nicht widersprechen müssen. Abstimmungen, wie sie in einigen Projekten vorgesehen sind, sind meist nur Mittel, um ein Stimmungsbild zu erzeugen.

Schafft es eine MaintainerIn in wichtigen Fragen nicht, einen Konsens herbeizuführen, so wird sie bald ohne TeilnehmerInnen da stehen. Gleichzeitig sind die TeilnehmerInnen darauf angewiesen, dass es Personen gibt, die die Aufgaben der MaintainerIn übernehmen und den Konsens organisieren. So ergibt sich eine gegenseitige Abhängigkeit zwischen MaintainerIn und anderen TeilnehmerInnen. Für das Produktivkraftmodell, das wir bei freier Software beobachten können, sind konsensorientierte Organisationsformen, wie z. B. das Maintainer-Modell, die Konflikte optimal ausbalancieren, unabdingbar.

7. Eine neue Qualität von Produktivkraftentwicklung

Eine wichtige Basis der Argumentation im Projekt Oekonux ist, dass es sich bei dem Phänomen freie Software um ein Beispiel für ein qualitativ neues Modell von *Produktivkraftentwicklung* handelt. Unter Produktivkraftentwicklung verstehen wir die historische Entwicklung der *Produktivkraft*, wobei Produktion in diesem Zusammenhang als das Stoffwechselverhältnis zwischen Mensch und (äußerer) Natur betrachtet wird. Produktivkraft fasst das Verhältnis zwischen Menschen, Produktionsmitteln und der Natur zusammen (Marx 1974).

Wir können drei Dimensionen von Produktivkraft unterscheiden. Die Dimension des *Inhalts* beschreibt, was der Inhalt menschlicher Tätigkeit ist – also die Art der Produkte, der Bezug zur Natur und die verwendeten Produktionsmittel. Im hier betrachteten Fall also freie Software und die für ihre Herstellung verwendeten Produk-

tionsmittel. Die Dimension der *Form* beschreibt die Art und Weise der Organisation des Produktionsprozesses – ob also z. B. Arbeitsteilung eingesetzt wird. Bei freier Software gehört die Selbstorganisation und das Maintainer-Modell in diesen Bereich. Die Dimension der *Produktivität* beschreibt die produzierte Gütermenge pro Zeiteinheit.

Wenn sich diese Dimensionen der Produktivkraft verändern, sprechen wir von einem qualitativen Schritt in der Produktivkraftentwicklung. Bei freier Software sehen wir eine Veränderung vor allem beim Inhalt und der Form der Produktivkraft. Im Folgenden werden einige Aspekte freier Software beschrieben, die wir für Hinweise auf diese neue Qualität halten.

Software insgesamt ist eine Produktgruppe, die erst seit einigen Jahrzehnten existiert. Ihre Nutzung setzt das Vorhandensein von Computern, mithin also hochmoderner Geräte voraus. Software, damit auch freie Software, ist also ein hochmodernes Produkt, das auf einem früheren Stand von Produktivkraftentwicklung gar keinen Sinn gemacht hätte. Freie Software befindet sich als Produkt an der *Spitze der allgemeinen Produktivkraftentwicklung*.

Nach wie vor unterliegen die Paradigmen, unter denen Software hergestellt wird, einem schnellen Wandel: Strukturierte Programmierung, Objektorientierung, Wasserfallmodell und agile Methoden – um nur ein paar zu nennen – haben sich innerhalb weniger Jahre abgelöst. Wir erleben ein Entwicklungstempo an den Wurzeln einer Technologie, das in anderen Ingenieursdisziplinen längst Vergangenheit ist. Mit einigem Recht kann freie Software als Produktionsweise ebenfalls als neues Paradigma bezeichnet werden. Einer der fundamentalen Unterschiede wird schon in der berühmt gewordenen Tanenbaum-/Torvalds-Debatte aus der Frühzeit der Linux-Entwicklung deutlich. Der Informatik-Professor Tanenbaum bezeichnete darin das von Torvalds für Linux avisierte Entwicklungsmodell als nicht praktikabel, da es nicht möglich sei, tausend Primadonnen zu kontrollieren. Die lakonische Antwort von Torvalds, die das Maintainer-Modell im Wesentlichen vorwegnimmt, bestand darin, dass es nicht seine Absicht sei, zu kontrollieren.[6] Diese Aufgabe von Kontrolle bezieht sich dabei sowohl auf die „Primadonnen" als auch auf den Code selbst. Freie Software gehört hinsichtlich seiner Produktionsweise zu einem der *innovativsten Ansätze*.

Freie Software wird nicht nur auf Computern benutzt und über das Internet verteilt, sondern auch mit Hilfe von Computern und Internet entwickelt. Computer allgemein und speziell ihre Anwendung in Form des Internets sind die zentralen Produktionsmittel für die Entwicklung freier Software. Diese Produktionsmittel gehören ebenfalls zu den *am weitesten entwickelten Produktionsmitteln*, die die Menschheit bisher hervorgebracht hat.

Im Gegensatz zu Produktionsmitteln vorangegangener Produktivkraftepochen sind Computer und das Internet auf Grund ihrer Universalität nicht auf Produktion digitaler Güter festgelegt, sondern können auch zum Spielen, zum Musikmachen, zum Diskutieren etc. eingesetzt werden. Die Produktionsmittel freier Software sind zunehmend *Teil der allgemeinen Infrastruktur* der sich am Horizont abzeichnenden Informationsgesellschaft.

6 Siehe hierfür http://www.educ.umu.se/~bjorn/mhonarc-files/obsolete/msg00089.html oder Torvalds (2001).

Diese allgemeine Infrastruktur ist heutzutage so preiswert und gleichzeitig allgemein nützlich geworden, dass sie in den hochindustrialisierten Regionen bereits beinahe überall verfügbar ist. Die Produktionsmittel, auf denen freie Software beruht, befinden sich also in *breiter privater Verfügung*. Auch dies ist ein Aspekt, der für Produktionsmittel vorangegangener Produktivkraftepochen nicht gilt.

Die Teilnahme an freien Software-Projekten ist nicht an Staaten oder Kulturkreise gebunden. Ganz selbstverständlich finden sich alle Interessierten an einem Projekt via Internet und kooperieren, um ein Produkt zu erstellen, das ihnen entspricht. Entwicklung freier Software ist *transnational*. Sie bezieht sich hinsichtlich ihrer Lizenzen auf die nationalstaatlichen Rechtssysteme der früheren Produktivkraftepoche und definiert somit einen eigenen Raum jenseits der Nationalstaaten.

Bemerkenswert ist auch, dass das gesamte Phänomen freier Software aus der Zivilgesellschaft kommt. Weder staatliche Agenturen noch Firmen haben freie Software hervorgebracht. Erst in neuerer Zeit, nachdem freie Software bereits erhebliche Erfolge erzielt hat, beginnen staatliche Einrichtungen und die Wirtschaft, auf den fahrenden Zug aufzuspringen. Freie Software würde sich auch unabhängig von diesen Einflüssen weiter entwickeln. Anstatt, dass Staat oder Wirtschaft die Kontrolle übernehmen, gibt es viele Beispiele dafür, dass sie sich den Gegebenheiten freier Software anpassen. So hat beispielsweise IBM zu Beginn seines Engagements im Bereich freier Software explizit darauf hingewiesen, dass man als großer Player behutsam mit der Community umgehen müsse – was allem Anschein nach bis heute erfolgreich durchgehalten wird und zu einer gewissen Reputation in der Community geführt hat. IBM hat verstanden, dass der Versuch eine Einschränkung der Freiheit die Kuh schlachten würde, die sie gerne melken möchte. So haben Firmen wie IBM ein großes Interesse daran, freie Software-Projekte zu unterstützen, da sie von den Leistungen der Community in höherem Maße profitieren, als sie es durch eine Kontrolle des Code erreichen würden.

8. Digitale Kopie als technologische Grundlage

Wir haben erläutert, dass bei der Entwicklung freier Software die Selbstentfaltung den individuell-sozialen Aspekt der Produktivkraftentwicklung bildet. Als technologische Seite dieser Entwicklung tritt die *digitale Kopie* hinzu. Während der Aspekt der Selbstentfaltung als gesellschaftliche Potenz schon immer da gewesen ist, handelt es sich bei der digitalen Kopie um eine historisch neue Potenz. Erst durch diesen technologischen Fortschritt ist die Ausdehnung der Selbstentfaltung als Grundlage eines Produktivkraftmodells möglich geworden.

Die digitale Kopie, die Möglichkeit also, digitale Informationen zu reproduzieren, hat im technologischen Sinne einige Eigenschaften, die sie älteren Technologien gegenüber voraus hat.

Während analoge Reproduktionen von Information immer mit Verfälschungen zu kämpfen haben, liefert die digitale Kopie eine *exakte Reproduktion* des Originals: Original und Kopie sind nicht zu unterscheiden. Mit diesem technologischen Fortschritt werden Begrenzungen der Verfügbarkeit digitaler Informationen nachhaltig beseitigt.

Zwei weitere Tatsachen moderner Technologieentwicklung geben der digitalen Kopie aber erst richtig Sprengkraft. Einerseits ist diese Reproduktionstechnik nämlich mittels Computern für sehr viele Menschen täglich und selbstverständlich verfügbar. Diese *breite Verfügbarkeit* führt dazu, dass die digitale Kopie kaum noch Einschränkungen unterliegt. Digital-Rights-Management-Technologien sind so gesehen nichts anderes als der (krampfhafte) Versuch, diese basale Eigenschaft moderner Technologie wieder zurückzunehmen, denn tatsächlich sind die beiden fundamentalen Operationen von Computern die Manipulation und die Kopie von digitalen Daten.

Andererseits steht mit dem Internet, das nichts anderes als eine planetenumspannende Fernkopiereinrichtung ist, eine Einrichtung zur Verfügung, die es ermöglicht, dass Informationsgüter auf einfachste Weise *global* verfügbar gemacht werden können. Das Internet verbindet die individuelle Verfügung über Informationsgüter mit dem allgemeinen Zugang zu ihnen.

Ein weiterer, eher subtiler Aspekt digitaler Kopie ist ihre *Universalität*: Der Inhalt, die Bedeutung des zu kopierenden Informationsguts, ist für den Vorgang der digitalen Kopie völlig unerheblich. Texte, Bilder, Musik, Programme können mit der gleichen Technologie reproduziert werden, sobald sie als Bytestrom vorliegen. So, wie die Kraftmaschinen der industriellen Ära (Dampfmaschine, vor allem aber Elektromotor) eine Basistechnologie für beliebige Anwendungen mechanischer Kraft und damit für die Industriegesellschaft bilden, so bildet die digitale Kopie eine Basistechnologie für die Informationsgesellschaft.

Auf dieser Grundlage ist das Internet von Beginn an auch als Kommunikationsmittel genutzt worden. Wie keine Kommunikationseinrichtung zuvor ermöglicht das Internet *globale Kommunikation* in Echtzeit. Es ist jetzt möglich, dass Menschen mit gleichen Bedürfnissen unabhängig von ihrem Standort in dem Tempo kommunizieren, das ihnen und ihrer Tätigkeit angemessen ist. Diese Kooperationsmöglichkeit ist wie die digitale Kopie selbst eine unabdingbare Voraussetzung für die Entfaltung freier Software.

9. Freie Software als Keimform

Eine der zentralen und nicht unumstrittenen Thesen im Projekt Oekonux ist die These von der freien Software als *Keimform einer neuen Gesellschaft* (Vgl. Kurz 1997). Unter einer Keimform verstehen wir ein Phänomen, das in den Rahmen eines bestehenden Gesamtsystems eingebettet ist, gleichzeitig aber Eigenschaften hat, die über die Logik des umgebenden Gesamtsystems hinausgehen und eine mögliche, neue Entwicklungsrichtung des Gesamtsystems darstellen können. Eine Keimform ist dabei noch keine vollständig entfaltete Form einer Gesellschaft, sondern zeigt nur einige identifizierbare Merkmale.

Eine neue Form entfaltet sich dagegen in mehreren Schritten. Das so genannte *Fünfschritt-Modell*, aus dem der Begriff Keimform stammt, erfasst in allgemeiner Weise, wie es innerhalb von Entwicklungsprozessen zu qualitativen Übergängen kommt. Es erklärt, wie Neues entsteht und sich schließlich durchsetzen kann. Das Fünfschritt–

Modell kommt ursprünglich aus der Kritischen Psychologie (Holzkamp 1983), aus der Analyse qualitativer Entwicklungsschritte in der Evolution. Die fünf Schritte sind:

1. Entstehung der Keimform
   Alles, was es selbstverständlich und allgegenwärtig gibt, ist irgendwann einmal etwas Neues, ganz und gar nicht Selbstverständliches gewesen. Über mehrere Schritte hat sich das Neue schließlich durchgesetzt. Dieses Neue, das später einmal Altes sein wird, nennt man Keimform. Keimformen können in Nischen und Sonderbereichen entstehen. Sie leben vom und im Alten, besitzen aber schon Formen des Neuen.

2. Krise der alten Form
   Keimformen erlangen nur Bedeutung, wenn das Alte in die Krise gerät. Das Alte kann im Wesentlichen aus zwei Gründen in die Krise geraten. Zum einen können sich äußere Bedingungen so dramatisch oder so schnell verändern, dass das alte Prinzip darauf nicht mehr angemessen reagieren kann. Zum anderen kann sich das Alte selbst erschöpft haben, wenn alle Entwicklungspotenzen ausgereizt sind. Stagnation wäre eine Reaktionsform, Zerfall eine andere.

3. Keimform wird zur wichtigen Entwicklungsdimension
   Unter den Bedingungen der Krise des Alten kann die Keimform die Nischen verlassen und sich quantitativ ausbreiten. Sie wird zu einer wichtigen Entwicklungsdimension innerhalb der noch dominanten alten Form. Diese Etablierung der Keimform kann zwei Richtungen einschlagen: Sie führt zur Integration in das Alte und zur Übernahme der alten Prinzipien oder die Keimform behauptet sich auf Grund der neuen Prinzipien immer besser im und neben dem Alten. Im ersten Fall geht der Keimformcharakter verloren, im zweiten Fall wird das Neue gestärkt. Das Alte kann in beiden Fällen von einer integrierten oder gestärkten Keimform profitieren und Krisenerscheinungen abmildern.

4. Keimform wird zur dominanten Größe
   Die frühere minoritäre Keimform wird zur dominanten Form der Entwicklung. Das Neue setzt sich durch, weil es hinsichtlich einer wichtigen Dimension des Gesamtprozesses besser ist. Damit endet der Keimformcharakter des Neuen. Nun sind seine Prinzipien bestimmend und verdrängen nach und nach oder auch schlagartig die überkommenen, nicht mehr funktionalen Prinzipien des Alten. Das Neue wird das selbstverständliche Allgegenwärtige.

5. Umstrukturierung des Gesamtprozesses
   Schließlich strukturieren sich alle Aspekte des Gesamtprozesses in Bezug auf das bestimmende, jetzt selbstverständliche Neue hin um. Das betrifft vor allem auch solche Prozesse, die im Gesamtprozess nicht bestimmend, sondern nur abgeleitet sind. Mit diesem Schritt ist nun potenziell wieder der erste Schritt eines neuen Fünfschrittes erreicht: Keimformen können auftreten, das dann alte Neue gerät in die Krise usw.

Alle Phasen können über kürzere oder längere Zeiträume andauern und es kann jederzeit Rückschritte geben. Nichts ist vorgegeben oder determiniert. Vollständig begriffen kann ein Fünfschritt der Entwicklung erst werden, wenn er vollzogen wurde und erst im Nachhinein kann man die frühere Keimform sicher identifizieren. Mitten im Entwicklungsprozess begriffen kann das Fünfschrittmodell helfen, die Sinne zu schärfen, um handlungsfähiger zu werden. Die umstrittene These im Projekt Oekonux lautet nun: „Bei freier Software haben wir es mit einer Keimform einer neuen Gesellschaft zu tun".

Wie beschrieben, zeichnet sich freie Software durch Wertfreiheit, Selbstentfaltung, Selbstorganisation und Globalität aus. Das alte Prinzip der Warengesellschaft basiert demgegenüber auf dem Wertgesetz, der Selbstverwertung, der Entfremdung und den Nationalstaaten. Die alte Form, die Warengesellschaft, ist erkennbar in der Krise. So versprach noch in den siebziger Jahren des vergangenen Jahrhunderts eine weitere Entfaltung und Vertiefung der Warengesellschaft nicht nur den Menschen der hoch industrialisierten Staaten individuelle Wohlstandsmehrung, sondern dieses Versprechen konnte auch erkennbar eingehalten werden. Von einem Fortschritt in dieser Hinsicht spricht heute niemand mehr. Vielmehr wird im Rahmen von Arbeitslosigkeit und Sozialabbau ganz offen und in steigendem Tempo nur noch darüber diskutiert, wie die Senkung des Lebensstandards breiter Bevölkerungskreise auch der hoch industrialisierten Staaten weiter vorangetrieben werden soll. Demgegenüber hat sich freie Software als neue Form von Reichtum etabliert, das jenseits der Formen der Verwertung existiert. Umstritten ist, ob freie Software bereits eine wichtige Entwicklungsdimension innerhalb der alten Form (dritter Schritt) geworden ist oder sich noch in einer der früheren Phasen befindet.

10. GPL-Gesellschaft

In der Diskussion um die gesellschaftlichen Potenzen des Entwicklungsmodells, das sich nach der These in freier Software keimförmig zeigt, kam der Begriff der GPL–Gesellschaft auf (Merten 2000). Er bezeichnet eine mögliche zukünftige Gesellschaftsform, die auf den Prinzipien der Entwicklung freier Software beruht. Es ist aus verschiedenen Gründen nicht seriös möglich, ein detailliertes Bild einer solchen Vorstellung zu entwerfen. Entlang der Prinzipien der Entwicklung freier Software lassen sich aber einige Rahmenelemente feststellen.

Ein wichtiges Element der Entwicklung freier Software ist die Tatsache, dass die verwendeten Produktionsmittel vergleichsweise vielen Menschen Selbstentfaltung ermöglichen. Der innere Grund dafür ist, dass Computer als universelle, Information verarbeitende und programmierbare Maschinen *unendlich viele Freiheitsgrade* haben. Diese Freiheitsgrade können von Menschen zur Entfaltung ihrer individuellen Kreativität genutzt werden.

In einer GPL-Gesellschaft wäre diese Eigenschaft tendenziell auf alle Produktionsmittel übertragbar. Dies bedeutet, dass der Maschinenpark, den die Industriegesellschaft für die Nutzung unter den entfremdeten Bedingungen der Lohnarbeit hervorgebracht hat, umgearbeitet oder neu entworfen werden muss: Die Arbeit an

einem Fließband dürfte beispielsweise nur für die allerwenigsten Menschen zur Selbstentfaltung führen, weswegen sie endgültig verschwinden müsste.[7]

Auch die weitere und noch beschleunigte *Automatisierung von Arbeitsprozessen* ist ein Mittel, um maximale Selbstentfaltung zu gewährleisten. Arbeitsprozesse, die im Kern von Maschinen übernommen werden, müssen nicht mehr von Menschen erledigt werden. Sie können sich den Aufgaben zuwenden, die genuin menschliche Fähigkeiten erfordern und nicht von Maschinen übernommen werden können.

Wie wenige andere Beispiele macht freie Software sichtbar, dass Selbstentfaltung nicht sinn- und zweckfreies Tun sein muss, wie es uns die Freizeitindustrie weismachen will. Vielmehr ist das Ergebnis der Entwicklung freier Software ein Produkt, das für viele Menschen nützlich ist. Selbstentfaltung, wie sie in freier Software praktiziert wird, hat also nicht nur für das Individuum eine positive Funktion, sondern nutzt der gesamten Gesellschaft. In einer GPL-Gesellschaft hätten noch sehr viel mehr Tätigkeiten diesen Charakter der *unmittelbaren Verknüpfung von individuellem und gesellschaftlichem Nutzen*.

Nicht zu vergessen ist, dass existierende freie Software frei verfügbar ist. Diese Eigenschaft ist einerseits eine Folge des offenen Entwicklungsprinzips, das die maximale Inklusion aller Interessierten zum Ziel hat. Unter den Bedingungen der universellen digitalen Kopierbarkeit führt dies zusammen mit dem *Copyleft* tendenziell zu allgemeiner freier Verfügbarkeit der Produkte. Andererseits ist diese freie Verfügbarkeit auch Voraussetzung für die blühende freie Software-Landschaft, denn auch die EntwicklerInnen von freier Software setzen auf von anderen entwickelter freier Software auf.

Die freie Verfügbarkeit ist also sowohl Folge als auch Voraussetzung des gesamten Entwicklungsmodells. Diese enge Verschränkung wäre in einer GPL-Gesellschaft ausgedehnt auf alle Informationsgüter sowie auf materielle Güter.

In einer GPL-Gesellschaft wären folglich viele Einrichtungen der Arbeitsgesellschaft überflüssig. Wo Güter frei verfügbar sind, ist die Form der Ware nicht mehr zu halten, die davon lebt, dass Güter künstlich verknappt werden. Werden keine Waren mehr – wohl aber Güter – produziert, so ist auch kein Geld mehr notwendig, das die Vergleichbarkeit von Waren vermittelt: Wo Güter frei zur Verfügung stehen, ist der Tausch eines Guts gegen ein anderes zur überflüssigen Handlung geworden. Nicht zuletzt würde unter den Bedingungen der GPL-Gesellschaft Entfremdungspotential an vielen Stellen tendenziell abgeschafft. Die wichtigste Produktivkraft einer GPL-Gesellschaft wäre die menschliche Selbstentfaltung.

In einer GPL-Gesellschaft würde die Selbstentfaltung der Individuen zur unmittelbaren Voraussetzung für die Selbstentfaltung aller: Nur wenn sich die Individuen entfalten können, entstehen Produkte, die für alle nützlich sind. Gleichzeitig sind diese nützlichen Produkte und deren freie Verfügbarkeit die Grundlage für die individuelle Tätigkeit: Die Selbstentfaltung aller ist also auch die Voraussetzung für die Selbst-

7 Bereits in der Industriegesellschaft gibt es verschiedene Versuche, die Selbstentfaltung in die Produktion zu integrieren und so die Kreativitätsreserven der Lohnabhängigen zu mobilisieren. Allerdings können diese Versuche nicht die strukturelle Schranke der Entfremdung des Arbeitsprozesses überwinden (Vgl. auch Gorz 2004).

entfaltung der Individuen. Wir haben es mit einem sich selbst verstärkenden Prozess zu tun, der die langfristige Tragfähigkeit einer GPL-Gesellschaft in einem günstigen Licht erscheinen lässt.

11. Historische Schritte im Vergleich

Wenn wir davon ausgehen, dass freie Software ein Hinweis auf einen fundamentalen Schritt in der Produktivkraftentwicklung ist, so kann ein Vergleich mit dem letzten fundamentalen Schritt in der Produktivkraftentwicklung interessante Einsichten geben. Die letzte fundamentale Änderung war der Umbruch von den feudal geprägten Gesellschaften des Spätmittelalters zu den industriell geprägten Gesellschaften der Neuzeit mit Beginn der Aufklärung. Die gerade erfundenen Industriemaschinen erforderten für ihren Betrieb hinsichtlich Technik und Organisation von Menschen eine völlig neue Produktionsweise. Dies verwob sich mit sozialen und ideologischen Entwicklungen, wie beispielsweise der Idee der Nationalstaaten, sodass innerhalb eines historisch relativ kurzen Zeitraums diese Entwicklung der Produktivkräfte zu einer tiefgreifenden Umstrukturierung der Gesellschaft führte.

Einerseits wurden damals menschliche, tierische und einige wenige natürliche Kraftquellen (Wasser, Wind) durch moderne Kraftquellen (Dampf, Elektrizität) abgelöst. Andererseits wanderte das Know-how über die Arbeitsprozesse von den sie ausführenden Menschen in die mechanische Konstruktion der Maschinen, so dass menschliche Arbeitskraft nur noch als Ergänzung zu den Maschinen benötigt wurde. Das Ergebnis dieser Umstellung waren eine Fülle neuer, nützlicher Produkte sowie Großprojekte, die ohne Einsatz industrieller Methoden nicht denkbar gewesen wären. Eine unrühmliche Antriebsfeder war die industrielle Entwicklung von Militärtechnik.

Gleichzeitig strukturierte sich die Lebensweise der Menschen tiefgreifend um. Die Art und Weise, wie Menschen in der feudalen, subsistenzorientierten Produktionsweise ihr Leben organisiert haben, war für die industrielle Produktionsweise in vielerlei Hinsicht ungeeignet. Um nur ein Beispiel herauszugreifen: Der vorindustrielle Umgang mit Zeit war weitgehend an den Hell-/Dunkelphasen und den konkreten, unmittelbaren eigenen Notwendigkeiten orientiert. Dies ging so weit, dass in manchen Klöstern die Länge einer Stunde über den Jahreslauf variierte. Für die industrielle Produktion, bei der der Zeittakt durch die Maschinen vorgegeben wird, waren das völlig unbrauchbare Verhältnisse. Es soll nicht verschwiegen werden, dass diese Umstrukturierung der Lebensweise durchaus nicht immer freiwillig geschah, sondern in erheblichem Ausmaß auch mit dem Einsatz von Gewalt einherging.

Auch beim Übergang von der feudalen zur bürgerlichen Gesellschaftsform lassen sich Keimformen ausmachen. So kann beispielsweise das Handelskapital, das sich bereits im Frühmittelalter auszubilden begann, als frühe Keimform betrachtet werden, bei der sich der kapitalistische Umgang mit Geld entwickelte. Die frühindustrielle Textilindustrie war für diesen Schritt in der Produktivkraftentwicklung die Keimform, in der schon viele Formen der späteren Industriearbeit auftauchten. Insbesondere traten hier sowohl die standardisierte Massenproduktion als auch der massenhafte Einsatz von bezahlten Arbeitskräften auf.

Es lässt sich an diesem Beispiel auch sehr schön betrachten, wie die feudalen Strukturen, die alten dominanten Strukturen also, sich an diese Entwicklung anpassten und sie auch nutzten. Die gesamte Kriegsproduktion, die die Fürsten für die Führung ihrer Kriege brauchten, hatte sich von der subsistenzorientierten Produktion über die Jahrhunderte hin vollständig entbettet. Die Kriegsproduktion der späten Feudalherren, in der auch industrielle Formen relativ früh auftauchten, war nur mit Geld überhaupt zu bewerkstelligen. Das dafür notwendige Steuersystem trieb das Geldwesen weiter an, das sich später als eine entscheidende Grundlage nach-feudaler Gesellschaftsformen herausstellen sollte. Söldnerwesen sowie stehende Heere können als frühe Form bezahlter „Arbeitskraft" betrachtet werden.

Heute wissen wir, dass diese Entwicklung, die in frühen Keimformen bereits erkennbar war, zu einer grundlegenden Umwandlung der Gesellschaftsform geführt hat. Auch wenn die Fürsten die Aufklärung und die mit ihr verbundende, industrielle Produktionsweise teilweise begrüßten oder sogar vorantrieben, so stellte sich doch heraus, dass die feudale Produktionsweise mit ihrem Privilegiensystem und der Leibeigenschaft industrieller Produktion unangemessen war. Die industrielle Produktionsweise musste sich also neue Grundlagen schaffen, die durch Geldwirtschaft und freie Lohnarbeit gekennzeichnet waren. Auch die gesamte gesellschaftliche Organisation folgte nach und nach dieser Entwicklung der Produktivkraft – am sichtbarsten in der Gründung bürgerlicher Nationalstaaten.

Heute sind wir an einem Punkt angekommen, wo sich die mechanische Konstruktion von Maschinen immer weiter von einem konkreten Arbeitszweck ablöst. Industrieroboter, Fabber[8] etc. sind nicht durch ihre Konstruktion auf einen bestimmten Arbeitsprozess festgelegt, sondern ihre mechanische Konstruktion steckt nur noch den Rahmen ihrer Möglichkeiten ab. Die Produktion konkreter Gegenstände ist bei diesen Maschinen bereits Gegenstand von Software, womit das Know-how über die Arbeitsprozesse zum Informationsgut wird. Computer verwenden diese Informationsgüter als Programme in automatisierten Prozessen, sodass menschliche Energie und Kreativität nur noch dafür benötigt wird, die Informationsgüter selbst zu erstellen.

Die Nutzung von Industriemaschinen erforderte auf Grund ihrer Beschränkungen eine Anpassung der Menschen an die Notwendigkeiten der Maschinerie, was in vielerlei Hinsicht letztlich eine Unterwerfung bedeutete. Die Produktion von Informationsgütern ist hingegen ein kreativer Prozess, bei dem gerade die schöpferischen Qualitäten des Menschen gefragt sind, die durch Unterwerfung vernichtet werden. Zog die beginnende Industriegesellschaft eine Unterwerfung der Menschen nach sich, so erfordert die beginnende Informationsgesellschaft eine Freisetzung der unbeschränkten Selbstentfaltung von Menschen.

Während sich beim Übergang von den feudalen zu den bürgerlichen Gesellschaften der Schwerpunkt der Produktion von der Nutzung des Bodens zur industriellen Produktion materieller Güter verlagerte, so verschiebt sich der Schwerpunkt der

8 Ein Fabber (*digital fabricator*) wird auch als „Fabrik in einer Kiste" bezeichnet. Fabber erzeugen dreidimensionale Werkstücke direkt aus digitalen Daten, z. B. aus Kunststoff oder Metall. Fabber werden derzeit hauptsächlich im „Rapid Prototyping" zur Fertigung von Prototypen und Modellen verwendet.

Produktion beim Übergang in die *heraufziehende Informationsgesellschaft* auf die Produktion neuer Informationsgüter. Der Wechsel zur Industrieproduktion erforderte einen fundamentalen Wechsel in der Gesellschaftsform. Informationsgüter, die in fast allen Aspekten anderen Bedingungen unterliegen als materielle Güter, erfordern eine ebensolche Umstrukturierung. Eine grundlegende Änderung der Gesellschaftsform erscheint unabdingbar.

Literaturverzeichnis

Gorz, A. (2004), *Wissen, Wert und Kapital. Zur Kritik der Wissensökonomie*, Rotpunktverlag, Zürich.

Himanen, P. (2001), *The Hacker Ethic and the Spirit of the Information Age*, 1. Aufl., Random House, Vintage, UK.

Holzkamp, K. (1983), *Grundlegung der Psychologie*, Campus Verlag, Frankfurt am Main.

Kurz, R. (1997), 'Antiökonomie und Antipolitik. Zur Reformulierung der sozialen Emanzipation nach dem Ende des „Marxismus"', *Krisis* **19**. http://www.giga.or.at/others/krisis/r-kurz_antioekonomie-und-antipolitik_krisis19_1997.html.

Marx, K. (1974), *Grundrisse der Kritik der politischen Ökonomie*, Dietz-Verlag, Berlin/DDR. Rohentwurf 1857–1858.

Meretz, S. (2000*a*), 'GNU/Linux ist nichts wert – und das ist gut so!', http://www.kritische-informatik.de/?lxwertl.htm.

Meretz, S. (2000*b*), *Linux Co. Freie Software – Ideen für eine andere Gesellschaft*, AG SPAK Verlag, Neu-Ulm.

Meretz, S. (2004), 'Freie Software. Über die Potenziale einer neuen Produktionsweise', *Widerspruch* **45**.

Merten, S. (2000), GNU/Linux – Meilenstein auf dem Weg in die GPL-Gesellschaft, *in* 'Dokumentation LinuxTag 2000'. http://www.oekonux.de/texte/meilenstein/.

Raymond, E. S. (2000), 'The Cathedral and the Bazaar', http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Torvalds, L. (2001), *Just for Fun – Wie ein Freak die Computerwelt revolutionierte*, Carl Hanser Verlag.

Open Source – Die Rückkehr der Utopie?

CHRISTIAN F. GÖRLICH UND LUDGER HUMBERT

(CC-Lizenz siehe Seite 463)

> „You think you know when you learn, are more sure when you can write, even more when you can teach, but certain when you can program." (Perlis 1982)

Die hier getätigten Überlegungen wollen zunächst als Beitrag zur *Selbstaufklärung* der Open-Source-Bewegung verstanden werden. Unter Rückgriff auf eine differenzierende Begrifflichkeit und klassische Problemstellungen der Philosophie in ihrem besonderen Verhältnis zu den Einzelwissenschaften – hier insbesondere der Soziologie und Wissenschaftsgeschichte – sollen Fragehorizonte rekonstruiert werden, die Antworten erhoffen lassen, die dann in den Auseinandersetzungen um Open Source auch den Anspruch der Akteure auf die Mündigkeit zu stärken versprechen. Aus dem aufgewiesenen Fragehorizont wird in diesem Beitrag eine wissens- /wissenschaftssoziologische Problemstellung thematisiert, die Open Source als gesellschaftliches Subsystem in den Blick nimmt und im Anschluss an Merton (1972) über die leitenden Normen reflektiert. In einem Exkurs wird unter Bezug auf Hagstrom (1965) dem Ursprung der gegenwärtigen Diskussion um die Open-Source–Bewegung als einer Tauschgesellschaft nachgegangen. Abschließend wird mit Blick auf den Titel dieses Beitrages darüber räsoniert, welche Funktionen das Subsystem Open Source in Analogie zu ähnlichen Subsystemen, die sich um Kreativität formieren, für das gesellschaftliche Gesamtsystem wahrnehmen könnte. Es geht insbesondere auch darum, den Begriff einer „Informatischen Vernunft" zu etablieren und in seinem Verhältnis zur Allgemeinbildung zu bestimmen. Informatische Vernunft will nicht nur *instrumentelle Kenntnis* sein, Informatische Vernunft will in dem epochaltypischen Schlüsselbereich der „neuen technischen Steuerungs-, Informations- und Kommunikationsmedien" (Klafki 1991*a*, S. 59) den philosophischen Anspruch der Aufklärung wach halten.

1. Open Source – Fragestellungen im Kontext philosophischer Betrachtungen

Open Source als geistige Bewegung in einer sich zur Wissensgesellschaft wandelnden Welt philosophisch zu denken, eröffnet vielfältige Perspektiven und Fragestellungen. Die nachfolgende Aufzählung benennt einige Facetten:

- Open Source unter dem Blickwinkel des für jede Gesellschaft konstitutiven Eigentumsverständnisses (Vgl. Künzli 1986; Lutterbeck und Gehring 2004, Kap. 5, S. 331 ff.)
- Open Source unter dem Blickwinkel einer normengeleiteten Wissenschaftsentwicklung (ethischer und wissenschaftssoziologischer Zugang)
- Open Source unter dem Blickwinkel der Produktion: Arbeit und/oder Spiel (entfesselte Kreativität)
- Open Source unter dem Blickwinkel emanzipatorischer Entwicklungspotenziale

Der zuletzt genannte Blickwinkel benennt auch jenseits postmoderner Attitüden das eigene Erkenntnisinteresse. Wie in These 6 von Görlich und Humbert (2003, S. 90) ausgeführt, wird hier Philosophie nicht – wie noch bei Hegel – metaphysisch als Platzanweiserin[1] für die Einzelwissenschaften, sondern eher in Anlehnung an Habermas als Platzhalterin der Vernunft in den sich immer weiter ausdifferenzierenden und auch sich verheiratenden Einzelwissenschaften verstanden. Die Reflexion von Grundlagen und zentralen Begriffen und der nachdenkende Rückgriff auf klassische Problemstellungen fühlt sich dabei nach wie vor der Aufklärung verpflichtet. Unter „klassisch" verstehen wir in diesem Zusammenhang in Anlehnung an Luhmann, dass die in der Vergangenheit angesprochenen Probleme heute noch strukturell wirkmächtig sind, auch wenn wir sie heute in einer anderen Sprache oder bezüglich einer anderen (neuen) Materie reformulieren müssen. Und mit „Aufklärung" knüpfen wir an Kants bekannte Rede von der „Aufklärung als Ausgang aus der selbstverschuldeten Unmündigkeit" an. Angesichts des Patchwork-Charakters der Welt sollte dieses Festhalten an der Aufklärung nicht als naiv missverstanden werden, sondern eher ironisch im Sinne Richard Rortys. Ironisch meint hier eine ungläubige Haltung gegenüber jedem Anspruch auf etwas Letztgültiges. Rorty steht hier ganz in der Tradition von Nietzsches Behauptung, dass Wahrheiten nur Illusionen seien, „von denen man vergessen hat, dass sie welche sind." Sprache, Selbstbild, Kultur und Lebensform sind kontingent. Soweit ist Rorty Teil des postmodernen Diskurses, der Aufklärung als Blendwerk erscheinen lässt. Jedoch hier wendet sich Rorty vom postmodernen Diskurs ab. Rortys Leistung besteht darin, „einen postmodernen Standpunkt zu beziehen, ohne damit auch der politischen Resignation das Wort zu reden. Vielmehr bleibt Rorty ein leidenschaftlicher Verfechter der Aufklärung, hält allerdings deren *Vokabular*, das um den

1 Für personenbezogene Bezeichnungen in diesem Beitrag wird – abgesehen von Zitaten – für geschlechtsspezifische Bezeichnungen das generische Femininum gewählt. Männer mögen sich nicht ausgeschlossen fühlen.

Begriff einer universal gültigen Vernunft gruppiert ist, für „veraltet" – im Gegensatz zu Habermas. Rorty versteht sich als Vertreter einer bildenden und zugleich ironischen Philosophie: Bildend meint hier die Negation einer systematischen Philosophie und positiv: „ein Sich-bekannt-Machen mit anderen Kulturen, geschichtlichen Epochen, [. . .] ein fortwährendes Ins-Gespräch-Kommen mit Fremden, ein Neubefragen des Bekannten". (Breuer et al. 1996, S. 123). Weiter führen sie aus:

> „Diese Haltung bewahrt eine Art lächelnde Skepsis auch gegenüber den eigenen Träumen und Wünschen, die wohl der beste Schutz gegen aggressive Fundamentalisten und autoritäre Ideologien ist. Sie führt in eine gewissermaßen ästhetische Existenz, in der, mangels anderer Autoritäten, die Sorge um die eigene Autonomie und die private Vervollkommnung im Mittelpunkt stehen". (Breuer et al. 1996)

In diesem Sinne sind die folgenden Überlegungen als Beitrag zur Selbstaufklärung der Open-Source-Bewegung zu verstehen; implizit wird damit behauptet, dass es auch hier um die klassischen Themen der „Mündigkeit/Unmündigkeit, Autonomie und Kreativität" geht.

Als erste Facette in einer Reihe von weiteren geplanten Beiträgen wird hier Open Source als geistige und soziale Bewegung in der Tradition der Wissens- und Wissenschaftssoziologie in den Blick genommen. Von der Spiegelung der gegenwärtigen Situation an klassischen Fragestellungen in der Vergangenheit versprechen wir uns nicht nur eine differenzierende Wahrnehmung und Sprache, sondern angesichts der aufgeregten Diskussion auch ein wenig mehr Gelassenheit (Vgl. Barbrook 1998).

2. Das Phänomen Open Source oder „Was ist Open Source?"

Wer heute mit einer gewissen Distanz – gleichsam von außen wie eine Ethnologin – die Welt der Informatiksysteme und ihre leibhaftigen Akteure betrachtet, dem erscheint die Welt der informatischen Systeme gleichsam durch zwei recht unterschiedliche Ethnien besiedelt:

Zum einen finden sie Menschen vor, die an dem zweiten Medienumbruch teilnehmen, indem sie Hardware in der Regel zusammen mit Software kaufen und damit angesichts der behaupteten „permanenten Innovation" in eine Spirale des Konsums einsteigen. Da Informatiksysteme universal nutzbar sind, wenn geeignete Software zur Verfügung steht und Software ohne großen Aufwand kopiert und transportiert werden kann, ist die Nutzung *nicht gekaufter* Software an der Tagesordnung.

Auf der anderen Seite finden die Ethnologen Menschen – „Opensourcianer" –, die sich auf der Basis frei zugänglicher Quellcodes der kollaborativen Entwicklung von Software verschrieben haben – in den selteneren Fällen persönlich bekannt, in der Regel eher als *invisible colleagues*. Sie erscheinen als solche in ihren Ritualen Verwandte des Stammes der Wissenschaftlerinnen zu sein.

Wenn diese metaphorische Beschreibung in die richtige Richtung zielt – und davon gehen wir aus – ist es nahe liegend, sich für die weitere Analyse und Beschreibung der diesbezüglichen Aussagen der Wissens- und Wissenschaftssoziologie zu vergewissern.

Dabei ist wie in vielen anderen Bereichen auch hier durch den Zweiten Weltkrieg ein markanter Bruch zu verzeichnen. Während wissenssoziologische Fragestellungen zunächst auch im deutschen Raum zu Hause waren – man denke nur an Max Scheler oder Karl Mannheim, die ihrerseits wieder auf lange Traditionen zurückgreifen konnten – war die weitere Entwicklung eher durch amerikanische Wissenschaftler bestimmt, was nach dem Kriege zu einem Reimport alter Denkansätze führte.[2]

2.1. Open Source als Soziales Subsystem

Seit Luhmann kann heute nicht einfach und allgemein verständlich über Systeme geredet werden. Um dennoch verstanden zu werden, geben wir ein Definitionsangebot aus der Zeit vor Luhmann an. Norman W. Storer hat *soziales System* als eine stabile Folge von Interaktionsmustern definiert, „die sich um den Austausch eines qualitativ einzigartigen Gutes organisiert haben und von einem Satz gemeinsamer Normen geleitet werden, die die fortwährende Zirkulation dieses Gutes erleichtern". Da es in einer Gesellschaft nur eine begrenzte Anzahl qualitativ unterschiedlicher Güter von verbreitetem und andauerndem Interesse gibt, lässt sich mit Talcott Parson im Hintergrund ein überschaubarer Kreis von Subsystemen herausgliedern: das ökonomische, das politische, das religiöse und das Familiensystem. Vereinfachend und sicher erläuterungsbedürftig organisieren sich diese Subsysteme um die Güter Geld, Macht, Sinn und Liebe, ausgehend von einem verbreiteten und dauerhaften Drang zur Kreativität, der seine Befriedigung durch Austausch mit anderen findet. „Mit anderen Worten: Es gibt ein weit verbreitetes und anhaltendes Interesse an dem Gut *Reaktion auf Kreativität*, so dass es die Grundlage eines fünften sozialen Systems[3] innerhalb der Gesellschaft bilden kann" (Storer 1972). Über diese Bemerkung hinaus ist nicht nur an die Wissenschaft, sondern auch an die Künste oder auch an Open Source zu denken.

Dabei ist exkursartig ein möglicher Unterschied in der Art der Reaktion auf Kreativität, in der Art der Anerkennung der Leistungen von Wissenschaftlerinnen und Open-Source-Entwicklerinnen, zu reflektieren.

Anerkennung in wissenschaftlichen Kontexten schlägt sich nieder in der *(dokumentierten)* Wahrnehmung der eigenen Position. Dies hat zur Konsequenz, dass Wissenschaftlerinnen ein großes Interesse daran haben, dass ihre Veröffentlichungen zitiert werden. Dies kann zu Verzerrungen führen. Im Wissenschaftskontext wird allenthalben *zirkulär* zitiert, um das eigene Ranking zu erhöhen – es gibt *Ingroups*, die sich gegenseitig *unterstützen*, da sie beispielsweise in einem sehr spezialisierten Segment forschen (jeder kennt jeden). Die in den Zitationsindices häufig auftauchenden Personen sind nicht unbedingt fachlich herausragend. Neben konkreten personalen Abhängigkeiten kennt die Forschung den sogenannten Matthäus-Effekt: „Wer hat, dem wird gegeben".

In der Informatik findet die erfolgreiche Entwicklung von Informatiksystemen keinen direkten Eingang in Zitationsrankings. Weitere kritische Elemente lassen sich

2 Siehe hierfür Weingart (1972), aber auch Meja und Stehr (1982).

3 Die Autoren würden formulieren: „Grundlage von weiteren Systemen".

in den einschlägigen Darstellungen zur Problematik von Zitationsquantifizierungsversuchen in großer Zahl ausmachen. Exemplarisch seien hier Mattern (2002), Stock (2000) und Weingart et al. (1998) genannt. Zusammenfassend kann festgestellt werden, dass die quantitative Bestimmung der Leistung(-sfähigkeit) von Forscherinnen sehr kritisch betrachtet werden muss. Dennoch wird mit Hilfe der zunehmend öffentlich verfügbaren Rankings die vermeintliche Leistungsfähigkeit einer Forscherin dokumentiert.

Im Open-Source-Umfeld gibt es kein Ranking in dieser Form, als konstruktive Leistung wird primär „die Entwicklung von Code" anerkannt: funktionierender, wartbarer, verständlicher Quellcode, der möglichst auch gut dokumentiert ist und (im besten Fall – trotz der oben angeführten Richtungsentscheidung bei GNU/Linux) aktuellen Erkenntnissen der Fachwissenschaft genügt. Darüber hinaus werden Open-Source–Projekte mit der Zeit durchaus von mehreren Personen geleitet. Über die Lizenz werden die Beiträge aller an dem Projekt konstruktiv mitarbeitenden – inklusive der Entwicklungsgeschichte – dokumentiert. Diese Form der Anerkennung, bis hin zu der Erwähnung, dass Personen zur Identifizierung und Behebung von Fehlern beigetragen haben, ist Antrieb und Herausforderung zugleich.

Um diese Erkenntnis auf den Punkt zu bringen: Was funktioniert, lässt sich durchsetzen/setzt sich durch/wird durchgesetzt.

Das mit Open Source konkurrierende soziale Subsystem des Marktes von Angebot und Nachfrage steht hier nicht im Vordergrund unseres Interesses, es dient eher als Hintergrundfolie, um die Eigentümlichkeiten der Open-Source-Bewegung umso schärfer zu konturieren. Auch die Nutzerinnen von Informatiksystemen nutzen diese Systeme letztlich, um einer gesellschaftlich geforderten Zielvorstellung zu entsprechen. Damit hoffen sie, als Gewinner in dem zweiten Medienumbruch dazustehen, gleichsam als *global player* in der Welt der informatischen Systeme. Angesichts solch hoher Zielvorgaben und faktisch knapper finanzieller Ressourcen beginnen Menschen zu *organisieren*, es kommt zu der von Merton beschriebenen Anomie in Subsystemen der Gesellschaft, wenn nicht in der Gesellschaft überhaupt (Vgl. Merton 1972).

Offensichtlich werden die Akteure der Open-Source-Bewegung von anderen Normen begleitet, von Normen, die in ähnlicher Weise auch das System der Wissenschaft steuern.[4] Nachdem lange in idealistischer Denktradition der Zusammenhang des Wissens mit der Sozialstruktur geleugnet wurde, hat Merton erstmals einen, die *scientific community* konstituierenden Normenkatalog mit dem Anspruch eines Ethos der Wissenschaft aufgestellt, der in der Folgezeit ergänzt, variiert, aber in der Substanz unangefochten blieb. Die von Merton aufgestellten Normen sind:

- der Universalismus
- der organisierte Skeptizismus
- der Kommunismus
- die Uneigennützigkeit

4 Auf die nötige Abgrenzung von Begriffen wie Norm, Werte o. ä. muss hier verzichtet werden. Wir wählen den Begriff Norm, um einen stärkeren Handlungsbezug hervorzuheben.

Diese Begriffe werden wegen ihrer möglichen Missverständlichkeit erläutert und auf die Open-Source-Bewegung bezogen.

Unter *Universalismus* versteht Merton als Kind seiner Zeit, die durch den Ost-West–Konflikt geprägt war, die Annahme, „dass Wahrheit und Wert einer wissenschaftlichen Aussage von den Charakteristika ihres Autors unabhängig sind" (Storer 1972, S. 62). So dürfte diese Aussage auch erkenntnistheoretisch heute nicht mehr haltbar sein, man betrachte nur die Nachbeben des Konstruktivismus. Andererseits ergibt sich für Open Source hier eine interessante Fragestellung: Zeigt die Open-Source-Bewegung eine Tendenz zur Universalisierung? Lassen sich in einer dialektischen Gegenbewegung kulturspezifische Regionalisierungstendenzen feststellen? – ein nach Hermann Lübbe immer wieder zu beobachtender Motor der Geschichte. Wir denken hier beispielsweise an die unterschiedlich sich entwickelnden GNU/Linux-Distributionen von SuSE (vormals deutsche Distribution), Red Hat Fedora (amerikanische Distribution), Mandrake (französische Distribution), Red Flag Linux (chinesische Distribution).

Unter *organisiertem Skeptizismus* versteht Merton, „dass jeder Wissenschaftler selbst verantwortlich gemacht wird zu prüfen, dass frühere Forschungen anderer, auf denen seine Arbeit aufbaut, richtig sind", er sieht damit auch die Pflicht verbunden, Kritik an der Arbeit anderer zu veröffentlichen, wenn der Wissenschaftler diese Arbeit für falsch hält (Vgl. Storer 1972, S. 63). Auch hier gilt es mit Blick auf Open Source, differenzierende Fragen zu stellen: Wenn die verschiedenen Subsysteme nicht nur über verschiedene Güter, sondern auch durch eine unterschiedliche Semantik gekennzeichnet sind, stellt sich die Frage nach der der Open-Source-Bewegung eigenen Semantik und zwar sowohl hinsichtlich der kognitiven als auch der sozialen Dimension. So ist das Wissenschaftssystem üblicherweise durch die Semantik „richtig/falsch" gekennzeichnet. Diese Attribuierung ist unter Berücksichtigung der genannten Relativierung von Wahrheitsansprüchen bei der Affinität zur Mathematik und den Naturwissenschaften auch auf die Open-Source-Bewegung zu übertragen. Das in der Open-Source-Bewegung produzierte Wissen hat darüber hinaus andere binär zu beurteilende Qualitäten: funktional/nicht funktional oder sogar im ästhetischen Sinne: schöne und hässliche Qualitäten. Beispiele für solche Beurteilungen sind:

– Zu der Kategorie „richtig/falsch" kann als Illustration die Auseinandersetzung zwischen Linus Torvalds und Andrew Tanenbaum betrachtet werden: Der Entwurf von GNU/Linux entspricht zum Zeitpunkt der ersten Planung nicht etwa dem aktuellen Stand der Betriebssystemforschung, wie Tanenbaum in der Diskussion deutlich zum Ausdruck bringt. Er kann Linus allerdings nicht davon abhalten, das Betriebssystem in dieser Weise zu konzipieren (Vgl. Tanenbaum und Torvalds 1992). Inzwischen zeichnen sich die Grenzen monolithischer Betriebssysteme zunehmend deutlich ab und führen zu Strategien, das Betriebssystem stärker zu modularisieren.[5]

5 Es stellt sich allerdings die Frage, ob Linux als Betriebssystem existieren würde, wenn seinerzeit Linus nicht den *monolithischen Weg* gegangen wäre, denn: Es sollte ein Betriebssystem entwickelt werden, kein Forschungsprototyp.

– In Mailinglisten zu Softwareprojekten gibt es häufig „Streit um den richtigen Weg". Die Auseinandersetzung wird beispielsweise dadurch ausgetragen, dass Quellcode ge*forkt* wird, d. h. aus bestehendem Quellcode wird eine neue Version ausgekoppelt. Eine erfolgreiche Variante hat die Chance, später wieder in den Hauptzweig aufgenommen zu werden. Die Entwicklung des ersten *Journaling File Systems* für Linux durch Hans Reiser hat die Entwicklung des *Extended File Systems 3*[6] kräftig befördert.[7]

– Gerade im Zusammenhang mit der Einführung neuer Funktionalität, die an gewisse Hardware gebunden ist, stellt sich immer wieder die Situation, dass mit der Methode *quick and dirty* Funktionalität verfügbar gemacht wird. Die so dazu gewonnene Funktionalität wird nach und nach in einem offenen Code-Review und der zunehmend sich verbreiternden Masse an Nutzerinnen sehr kritisch einem Test unterzogen, da es regelmäßig Nutzerinnen gibt, die gewisse Funktionen unbedingt nutzen wollen, auch wenn dies nur um den Preis der Aufgabe der unter GNU/Linux bekannten Stabilität möglich ist: Sie wollen nun mal beispielsweise ihren günstig erworbenen USB-Stick auch unter GNU/Linux nutzen können.

Unter *Kommunismus* oder *Kommunalität* versteht Merton, dass Forschungsergebnisse – in dem hier betrachteten Kontext also Softwareentwicklungen – frei und ohne Begünstigung anderen Wissenschaftlern – hier also Entwicklerinnen, aber auch Benutzerinnen als potenziellen Entwicklerinnen – mitgeteilt werden: „ [. . .] denn Wissen, das nicht an die Öffentlichkeit gelangt, kann nicht Teil anerkannten Wissens sein, an dem die Kreativität gemessen wird und auf das andere Wissenschaftler (Entwickler/-Nutzer) sich in ihrer Arbeit beziehen" (Merton lt. Storer 1972, S. 64). Freie Mitteilung der Arbeitsergebnisse wird hier als eine entscheidende Voraussetzung für den Austausch von kreativen Programmen und Anerkennung dieser Kreativität angesehen. Auch hier wird mit Blick auf die Differenzierung von Quellcode und Weiterentwicklungen nachzufragen sein: Es gibt auch in anderen Bereichen eine kommerziell umarmte Kreativität. Wer möchte den Hobbymalerinnen, die ihre Farben nicht selber herrichten, sondern im Baumarkt kaufen, die Kreativität absprechen?

Im Unterschied zu Hobbymalern sind alle Entwicklerinnen von Open-Source-Bausteinen abhängig von den (funktionierenden) Ergebnissen vorgängiger Entwicklerinnen und damit gezwungen, in einem Netzwerk konstruktive Leistungen kollaborativ zu entwickeln. Selbst wenn diese Aktivitäten scheinbar isoliert stattfinden, so gilt der Wahlspruch „standing on the shoulders of giants". Sehen wir uns die quantitative Komplexität moderner Informatiksysteme näher an. Es wird quasi parallel sowohl

6 *ext3* gehört zu der Kategorie der Journaling File Systeme.

7 Unter http://www.kerneltraffic.org/kernel-traffic/kt20000710_75.html#1 findet sich die Auseinandersetzung über die Integration von ReiserFS in den Linuxkern. Unter http://www.kerneltraffic.org/kernel-traffic/kt20000103_49.html#3 verdeutlicht die Diskussion um die Entscheidungsfindung, welches Dateisystem Eingang in den Linuxkern finden soll. Die Diskussion um die konstruktive Weiterentwicklung ist nicht abgeschlossen, wie unter http://www.kerneltraffic.org/kernel-traffic/kt20030910_231.html#13 nachgelesen werden kann:
„At this point other folks came into the discussion, and the thread ended inconclusively."

am Fundament, aber auch an vielen Dachgauben gearbeitet, eine neue Straße gebaut, nebenbei arbeiten einige auch noch unterhalb des Hauses, weil dort Gold oder Öl vermutet wird. Solcherart Arbeiten führen zu maximaler Kooperationsnotwendigkeit – verteilte Teamarbeit, Code-Review, Kohorten von Personen, die bereit sind, die Stabilität ihrer Systeme aufzugeben, um neue Funktionalitäten auf ihre Verträglichkeit und Stabilität zu testen, Gruppen von Personen, die die Dokumentation weiterentwickeln. Und all dies verteilt über den gesamten Planeten Erde. Ein interessanter Aspekt bei dieser Art der Arbeit ist die Art der Verantwortlichkeit, die jeder Beteiligten für *das Ganze* zukommt. Da niemand diese Komplexität „beherrschen" kann, wird permanent konstruktiv „gestritten", es werden Entscheidungen getroffen, um die sprichwörtliche Stabilität zu erzielen, die Linux gegenüber proprietären Systemen auszeichnet.

Unter *Uneigennützigkeit* versteht Merton (nach Storer 1972, S. 64) eine Norm, die die „Wissenschaft um ihrer Selbst willen bestärken, Wissenschaft und Forschung zum Selbstzweck machen will." Dies mag idealistisch klingen und steht in einem scheinbaren Widerspruch zu dem Tausch der bereitgestellten kreativen Leistung gegen Anerkennung.[8]

Es dürfte deutlich geworden sein, dass diese Überlegungen eher differenzierende Fragen aufwerfen als Antworten geben wollen. Überleitend zur Frage der gesamtgesellschaftlichen Bedeutung der Open-Source-Bewegung sei an die wechselnde Wertschätzung erinnert, die die Beschreibung einer Gesellschaft als Tauschgesellschaft erfahren hat.

Unter dem Titel „Gift-Giving as an Organizing Principle in Science?" (Hagstrom 1965), zitiert in Barnes (1972) hat Warren O. Hagstrom diese Tauschmechanismen vor dem Hintergrund einer Befragung von Wissenschaftlern genauer beschrieben.

> „Manuscripts submitted to scientific periodicals are often called *contributions* and they are, in fact, gifts. Authors do not usually receive royalties or other payments, and their institutions may even be required to aid in the financial support of the periodical. On the other hand, manuscripts for which the scientific authors do receive financial payments, such as textbooks and popularizations, are, if not despised, certainly held in much lower esteem than articles containing original research results [. . .] "

Diese Aussage hat nach unserem Eindruck auch heute noch in weiten Bereichen der Wissenschaft ihre Gültigkeit. Hagstrom legt in Anlehnung an Smelser nahe, dass es beim Tausch immer auch um mehr geht:

> „ [. . .] that the gift mode of exchange is typical not only of science but of *all institutions concerned with the maintenance and transmission of common values, such as the family, religion, and communities.*[9] In general, the acceptance of a gift by an individual or a community implies a recognition of the status of the donor and the existence of certain

8 In der Tat wäre eine hier nicht zu leistende Rückbesinnung auf idealistische Reflexionen über das Reich der Notwendigkeit, der Schönheit und Freiheit etwa von Friedrich Schiller möglicherweise hilfreich.

9 Hervorhebung durch die Autoren

> kinds of reciprocal rights. These reciprocal rights may be to a return gift of the same kind and value, as in many primitive economic systems, or to certain appropriate sentiments of gratitude and deference. In science, the acceptance by scientific journals of contributed manuscripts establishes the donor's status as a scientist indeed, status as a scientist can be achieved only by such gift-giving—and it assures him of prestige within the scientific community."

Damit stehen wir vor der Frage, für welche Werte[10] die Open-Source-Bewegung steht. Welche wechselseitigen Verpflichtungen werden von ihren Akteuren erwartet?

Unter dem Stichwort „intrinsically gratifying" thematisiert Hagstrom neben der soziologischen auch eine psychologische Dimension dieses Austausches. Er ist offensichtlich fest davon überzeugt: „Most scientists have sincere interests in the advancement of science, more than in their own recognition." Diese idealistisch anmutende These dürfte je nach anthropologischen Vorannahmen kontrovers gesehen werden. Gleichwohl wäre es interessant, die Akteure der Open-Source-Bewegung einmal zu diesem Aspekt zu befragen.

Hier soll das Augenmerk auf den Vergleich mit *many primitive economic systems* gelenkt werden. Die Semantik von *primitive* ist ambivalent besetzt, negativ im Sinne von roh, unkultiviert, andererseits aber durchaus positiv im Sinne von grundlegend oder in der Kunst im Sinne von frühe Meister. Diese semantische Ambivalenz verweist auch auf unterschiedliche philosophische Sichtmöglichkeiten. Während im Kontext meist eurozentrierter Fortschrittsphilosophien und -ideologien das Frühere oft auch das Primitivere und Unkultiviertere war, haben wir unter dem Einfluss des Strukturalismus gelernt, Naturvölker weniger als Vorform, sondern als äquivalente Lebensentwürfe zu lesen und den Genozid und „Kulturozid" der letzten fünf Jahrhunderte als Verlust zu begreifen.[11]

2.2. Exkurs über die Konkurrenz

Die Open-Source-Bewegung wird in der Öffentlichkeit vor allem und zuerst als David im Konkurrenzkampf, mit dem auf Klientenseite marktbeherrschenden Giganten,[12] in den Blick genommen.

Aufgeregtheiten und Grabenkämpfe deuten auf Unverständnis und Neues hin. Die Bewahrenden verteidigen ein erfolgreich scheinendes und funktionierendes Paradigma gegen eine Weltsicht, die zuerst einmal bekannte Sicherheiten und Geschäftsmodelle

10 *maintenance and transmission of common values*

11 Wenn wir das schwierige Problem einer quantitativen Erfassung einmal beiseite lassen, so gab es zur Zeit der conqista 4 000 Kulturen auf der Welt, „die sich genügend voneinander unterschieden, um als eigenständig angesehen zu werden: [. . .] Heute, 500 Jahre später, hat sich die Zahl der Kulturen weltweit auf 500 reduziert [. . .] . Das bedeutet einen Verlust von 88% der kulturellen Vielfalt der Welt [. . .] " (Vgl. Galtung 1993, S. 11).

12 Vor der „Machtergreifung" auf PCs durch das Betriebssystem MS-DOS (und folgende, bis hin zu den kaum noch zählbaren Versionen von Windows) hieß dieser Gigant IBM, der sich zur Zeit – sicher nicht uneigennützig – nach dem ökonomischen Scheitern des technisch durchaus ausgereiften OS/2 für Linux engagiert.

in Frage stellt. Damit wird tatsächlich existierenden, profitablen *Seilschaften* mit einem noch nicht etablierten Weg (dem neuen Paradigma, das immer erst nachträglich als ein solches identifiziert werden kann) die Existenzberechtigung abgesprochen. Dies führt auf beiden Seiten tendenziell zu nicht rational erklärbaren Verhaltensweisen.

> „Es ist bemerkenswert, dass sich mit der Open-Source-Bewegung gerade in den Vereinigten Staaten als dem Inbegriff der kapitalistischen Welt ein Ansatz etabliert, der deutlich kommunistische Züge trägt. So stellen die Mitglieder der Open-Source-Bewegung das Ergebnis ihrer unentgeltlichen Arbeit der Allgemeinheit kostenlos zur Verfügung. Aus dieser Sicht verwundert es wenig, dass gerade Microsoft und dessen Gründer Bill Gates als Prototyp des Kapitalisten als Hauptfeindbilder fungieren. Folge dieser politischen Sicht der Open-Source-Bewegung sind zwei Hauptströmungen in der Bewegung. Zum einen gibt es die Ideologen, für die Open-Source-Software fast schon einer Weltanschauung gleicht. Auf der anderen Seite gibt es aber auch die Pragmatiker, denen wohl vor allem die breite Diffusion der Open-Source-Software in den letzten Jahren zuzuschreiben ist." (Acker und Hettich 2001, S. 2)

Wird diese sichtbar gewordene Konkurrenz nicht nur als eine Sichtweise eines ökonomischen Sachverhalts abgetan, sondern philosophisch-gesellschaftswissenschaftlich vertieft, stellt sich die Frage nach der Bedeutung der Konkurrenz für einen gesellschaftlichen Produktionsprozess, der immer auch ein geistiger ist. „Die Bedeutung der Konkurrenz im Gebiete des Geistigen" lautet der Titel eines Vortrages, den Karl Mannheim 1928 auf dem Sechsten Deutschen Soziologentag in Zürich hielt (Vgl. Mannheim 1928). Dieser Vortrag kann in dem oben angegebenen Sinne von Luhmann als klassisch angesehen werden. Heute würden wir wohl eher von der „Bedeutung der Konkurrenz auf dem Gebiet des Wissens" sprechen. Der Rückgriff auf Mannheims Kategorisierung verspricht eine zunächst verfremdende Sichtweise und eröffnet weitere Fragemöglichkeiten auf dem Wege zur Selbstaufklärung der Open–Source-Bewegung.

Nach Mannheim ist die Konkurrenz nicht ein Phänomen der ökonomischen Sphäre, sondern genuin ein Phänomen des gesamten gesellschaftlichen Lebens. Der Begriff Konkurrenz wird nicht aus der Sphäre des Ökonomischen heraus verallgemeinert, sondern umgekehrt, „als die Physiokraten und Adam Smith die bedeutende Rolle der Konkurrenz im Ökonomischen aufwiesen, entdeckten sie nur eine allgemeine soziale Beziehung im besonderen Elemente des Ökonomischen" (Mannheim 1928, S. 332). Der Konkurrenz kommt also auch für den hier thematisierten gesellschaftlichen Bereich des Wissens eine konstitutive Bedeutung zu – in dem Sinne, dass sie nicht nur „peripher als Antrieb, als Anlass, als Gelegenheitsursache zur geistigen Produktion da ist [. . .], sondern dass ihre jeweilige Form konstitutiv in die Gestalt und in den Gehalt der Kulturobjektivationen und in die konkrete Form der Kulturbewegung hineinragt" (Mannheim 1928, S. 328). Mit seinem wissenssoziologischen Ansatz sieht sich Mannheim in der Nachfolge phänomenologischer und geschichtsphilosophischer Traditionen. Neben der Dynamik und der Morphologie des Wissens

wird bei ihm die konstituierende Bedeutung des historischen Zeitmoments in das Zentrum der Reflexion gerückt. Nicht weiter verfolgt wird hier seine Differenzierung zwischen *seinsverbundenem Denken* und *Bewusstsein* überhaupt. Das Wissen, das sich in der Open-Source-Bewegung artikuliert, wird von uns als informatisches Wissen dem seinsverbundenen Wissen zugeordnet.

Damit eröffnet sich der folgende Fragehorizont:

> Welche Form der Konkurrenz wirkt in welcher Weise konstitutiv in die Gestalt und den Gehalt von Open Source als einer Kulturobjektivation und -bewegung hinein?

Zur weiteren Klärung charakterisiert Mannheim Konkurrenz als Wettbewerb, der unter anderem durch verschiedene Parteien mit gleichen Zielsetzungen gekennzeichnet ist und der tendenziell in einen Konflikt, wenn nicht gar Kampf, aber auch in ein Miteinander münden kann. Diese Charakterisierung verweist indirekt auf die Bezeichnung für ein aktuelles quelloffenes Produkt zur kollaborativen Entwicklung von Software: *subversion* (Collins-Sussman et al. 2004).[13]

Bezüglich der Zielsetzung sei auf die vielfach kritisierte Haltung von Linus Torvalds hingewiesen, die durch den autobiographischen Beitrag „Just For Fun" von Diamond und Torvalds (2002) bewusst nicht ideologisch überhöht, sondern in Form eines Understatements darstellt, dass die Bewegungsmomente keiner Ideologie verpflichtet scheinen. Dass in der *Community* allerdings auch ein anderer Wind weht, zeigt sich in der permanent stattfindenden Auseinandersetzung um die verschiedenen Lizenzmodelle. So wird [La]TEX zur Zeit nicht auf einer der *freien* Entwicklungsplattformen weiterentwickelt, weil die LATEX-Lizenz (LPPL) bisher nicht durch die *Open Source Initiative* zertifiziert wurde. Andererseits ist LATEX eines der ältesten und erfolgreichsten offenen Projekte (Vgl. Schröder 2004).

Mit Blick auf Open Source konnten die verschiedenen Parteien als David und Goliath in einer mythischen Interpretation benannt werden, bis große Unternehmen (allen voran IBM) damit begannen, die *Bewegung* zu unterstützen, um Geschäftsinteressen zu verfolgen. Luthiger verdeutlicht die Dimensionen, die zur Mitarbeit in offenen Projekten führen (vgl. Luthiger 2004). Ein erfolgreiches Softwareprodukt (StarOffice) wurde von der Firma Sun als Open Office unter eine Open-Source-Lizenz gestellt. Darüber hinaus beginnt die Firma Microsoft 2004, erste Codeschnipsel unter einer Open-Source-Lizenz zu veröffentlichen. Hinsichtlich der Zielsetzungen wird hier nun eine These aufgestellt, die die philosophische Dimension des oben angegebenen Fragehorizonts verdeutlicht:

> Den konfligierenden Parteien geht es – von außen betrachtet – darum, ihre jeweilige Weltauslegung zur öffentlich herrschenden Weltauslegung zu machen.

Gegen diese analytische Zuschreibung spricht die Erfahrung mit den vielen verschiedenen Lizenzmodellen der Open-Source-Gemeinde (z. B. GPL, BSD, MPL,

13 Weitere Informationen unter (http://subversion.tigris.org/).

QPL, LPPL).[14] Nicht umsonst gibt es sehr viele, in ihren Details sehr unterschiedliche Open-Source-Lizenzen, häufig ändern Autoren gewisse Kleinigkeiten in den Lizenzbedingungen, die sie subjektiv für wichtig halten. Damit wird deutlich, dass es nicht die eine Linie zwischen proprietär und offen gibt. Auch das macht es „den Proprietären" so schwer, mit offenen Produkten „umzugehen". Wir befinden uns eher in einem Zustand der permanenten Diversifizierung, da jede (freie) Autorin die Lizenz, nach der ihr Werk verfügbar ist, nach individuellen Kriterien formulieren kann. Will eine Autorin allerdings Ressourcen nutzen, so muss sie prüfen, ob ihr Lizenzmodell unterstützt wird.

Mannheim beruft sich auf Heideggers „Man" und führt dessen Ansatz weiter, „dass die öffentliche Auslegung des Seins nicht einfach da ist, sie wird auch nicht ausgedacht, sondern es wird um sie gerungen" (Mannheim 1928, S. 335). Als Beispiel möge die Einflussnahme auf die sogenannten Offenen Standards des Internet durch die großen Konzerne dienen, die im W3-Konsortium inzwischen massiv ihre Interessen durchzusetzen versuchen. Auch die inzwischen in der Liste der Open-Source-Lizenzen auftauchenden Firmennamen stützen diese These.[15] Die entscheidenden Unterschiede in den hier thematisierten Weltauslegungen liegen in den jeweils unterschiedlichen Verständnis zentraler Kategorien wie beispielsweise der des „geistigen Eigentums", Arbeit und Kreativität.

Mag vordergründig von einer Offenlegung des Quellcodes gesprochen werden und scheinbar ein (Streit-)Gespräch über ein Weltsegment geführt werden, mit der Eigentumsfrage ist ein die Welt insgesamt ordnendes Prinzip genannt. Und gerade hier wird deutlich, dass die Autoren sich im Kontext der „Freien" in besonderer Weise ihrem Eigentum verpflichtet fühlen, in dem sie es offen zur Verfügung stellen. Dabei zeigt sich Autorenschaft von einer neuen Seite. Die bekannte Verwertungs- und Abhängigkeitskette wird durchbrochen.

Die Auslegung des Seins kann nach Mannheim (1928, S. 336) idealtypisch zustande kommen:

1. auf Grund eines Konsensus, bedingt durch eine spontane Kooperation der Einzelnen und Gruppen

2. auf Grund der Monopolsituation einer auslegenden Gruppe

3. auf Grund der Konkurrenz vieler Gruppen, die ihre besondere Seinsauslegung durchsetzen wollen

4. auf Grund der Konzentration mehrerer vorher atomisiert auftretender Konkurrenten zu einem Standorte, wodurch sich die Konkurrenz in der Gesamtheit auf wenige immer mehr herrschend werdende Pole reduziert

14 Eine Übersicht findet sich unter http://www.fsf.org/licenses/license-list.html.

15 So existieren von einigen bekannten Firmen eigene Lizenzen, wie z. B. Microsoft Shared Source License, Sun Public License, Nokia Open Source License, Apple Public Source License.

Diese Typen der „Auslegung des Seins" können auch als Typen der *Weltauslegung* von Open-Source-Gemeinschaften gelesen werden. Mannheim gibt hier formale Kategorien vor, die jeweils historisch zu situieren und materiell zu füllen sind.

Der Hinweis auf das Idealtypische impliziert, dass jede konkrete historische Situation als Mischform zu beschreiben ist. Während Mannheim hier in Jahrhunderten denkt, dürfte hinsichtlich Open Source in Jahrzehnten, wenn nicht in Jahren zu denken sein. So scheint gegenwärtig die Phase der Monopolstellung von Microsoft zumindest an den Rändern in Frage gestellt zu werden. Es mag als kühn erscheinen, hier Mannheims Beispiel für eine Monopolsituation heranzuziehen: die mittelalterlich-kirchliche Weltauslegung – grundsätzlich gekennzeichnet durch eine strukturelle Statik. Doch ein Vergleich drängt sich auf: Im Mittelalter waren es die geheiligten Bücher, die nur einer bestimmten Elite zugänglich waren und den *Sensibilitätskreis*, d. h. die Handlungsmöglichkeiten, beschränkten. Heute beschränkt ein nicht offen gelegter Quellcode den Sensibilitätskreis all jener, die eigentlich teilhaben und ihren Lebens- und Arbeitskreis kreativ gestalten und dynamisch weiterentwickeln wollen. Eine ganz besonders brisante Situation ergibt sich, wenn monopolisierte Informationstechnologien in ein allgemeines staatliches Bildungsmonopol implementiert werden. Es dürfte deutlich geworden sein, dass es sich bei der Entwicklung von Open Source also nicht nur um ein ökonomisches Problem, sondern eine für die Konstitution unseres kulturellen Selbstverständnisses zentrale Frage handelt. Die Frage könnte lauten: Erwächst in Open Source die Idee einer anderen Gesellschaft im Sinne der klassischen Utopien von Thomas Morus, Campanella, Bacon über Thoreau und Skinner bis zu den Träumen von 1968?

3. Open Source – die Rückkehr der Utopie?

1998 schreibt Richard Barbrook unter der Kapitelüberschrift „Das verlorene Utopia":

> „Das Netz wird von den enttäuschten Hoffnungen der 60er verfolgt. Weil diese neue Technologie eine weitere Periode schneller Veränderungen symbolisiert, blicken viele zeitgenössische Kommentatoren auf die abgestorbene Revolution von vor 30 Jahren zurück, um Erklärungen dafür zu finden, was jetzt gerade passiert." (Barbrook 1998)

Barbrooks Aufsatz macht trotz seiner polemischen Rhetorik deutlich, dass Open Source sich als soziale Bewegung eigentlich gegen zwei Konkurrenten behaupten muss: zum einen gegen die ökonomische, rechtliche und staatliche Gängelung, zum anderen gegen eine Vereinnahmung durch einen „aristokratischen Anarchismus", den er eng mit den Namen Deleuze und Guattari verbunden sieht. In seiner überzogenen Polemik gegen die „Deleuzoguattarianer" werden interessante Denkanstöße von Deleuze und Guattari vorschnell verschüttet.[16] Diesen wird andernorts nachzugehen sein (Vgl. Balke 1998, S. 116 ff.).

16 Gleichwohl weist Barbrook wohl berechtigt auf die Gefahren eines Elitarismus und Absolutheitsanspruchs hin. „Gefangen in ihren Glaubensregeln können die Schüler von Deleuze und Guattari nicht einmal erfassen, warum das Wachstum des Netzes wirklich ein so subversives Phänomen ist." Das

> „Von Anfang an hat die Geschenksökonomie die technische und soziale Struktur des Netzes bestimmt. Obwohl es vom Pentagon finanziert wurde, konnte das Netz nur erfolgreich weiterentwickelt werden, indem man den Benutzern erlaubte, das System für sich selbst zu erschaffen. Innerhalb der akademischen Gemeinschaft ist die Geschenksökonomie schon lange die Hauptmethode, Arbeit zu sozialisieren. Finanziert von Stiftungen oder dem Staat, veröffentlichen Wissenschaftler ihre Forschungsergebnisse, indem sie Vorträge halten und Artikel beisteuern. Trotz der verstreuten Art dieser Geschenksökonomie im Bildungsbereich erlangen Akademiker ihren intellektuellen Respekt voreinander durch Zitate in Artikeln und anderen Formen öffentlicher Anerkennung. Die Zusammenarbeit vieler verschiedener Wissenschaftler wird nur durch diese freie Verteilung von Information möglich." (Barbrook 1998)

Die Konvergenz zu unserer oben angeführten Argumentation ist deutlich, in Verbeugung vor der Open-Source-Bewegung nennt Barbrook ein weiteres überzeugendes Beispiel für die Kreativitätssteigerung:

> „Die Hi-tech Geschenksökonomie bildet sogar die Spitze der Softwareentwicklung. Zum Beispiel gibt Bill Gates zu, daß Microsofts größter Konkurrent, wenn es um das Zurverfügungstellen von Webservern geht, das Apache Programm ist. [. . .] Weil sein source code nicht durch Copyright geschützt ist, können Apache Server modifiziert, mit Zusätzen ausgestattet und von jedem verbessert werden, der die notwendigen Programmierkenntnisse besitzt. Sharewareprogramme beginnen jetzt, das Herzprodukt des Microsoftimperiums zu gefährden: das Windows Betriebssystem. Ausgehend von dem ursprünglichen Softwareprogramm von Linus Torvalds, baut jetzt eine Gemeinschaft von User-Entwicklern gemeinsam ihr eigenes, nicht gesetzlich geschütztes Betriebssystem: Linux. Zum ersten Mal hat Windows einen echten Konkurrenten." (Barbrook 1998)

Man muss darauf hinweisen, dass einige Formulierungen von Barbrook nicht ganz richtig sind: Apache ist keine Shareware. Der Quellcode ist sehr wohl geschützt, zumindest im Sinne der OS-Lizenz. Jedoch ist anzumerken, dass der Quellcode nicht im ökonomischen Sinne geschützt ist, er also für eigene Zwecke genutzt und verändert werden darf.

Im Gegensatz zu den *aristokratischen Anarchisten* spricht sich Barbrook für eine Koexistenz aus:

> „Gleichzeitig arbeiten Millionen von Menschen spontan miteinander im Netz, ohne die Koordination von Seiten des Staates oder des Marktes zu benötigen. Anstatt ihre Arbeit für Geld einzutauschen, schenken

Phänomenale liegt für Barbrook in der teilweisen Organisation des Netzes nach dem Muster einer Geschenksökonomie – frei von der Korruption der Konsumgesellschaft, aber auch frei von einem Totalitätsanspruch.

> sie ihre Kreationen im Austausch für freien Zugang zu Informationen, die von anderen produziert wurden. Diese Zirkulation von Geschenken koexistiert mit einem Austausch von Waren und Finanzierung durch Steuergelder. Wenn sie online sind, wechseln Menschen andauernd zwischen verschiedenen Formen sozialer Aktivität. Zum Beispiel kann ein User in einer Sitzung zuerst mit einem E-Commerce Katalog einkaufen, dann auf der Website der Gemeinde nach Informationen suchen und dann einem Listserver für Schriftsteller einige Gedanken beitragen. Ohne auch nur bewußt darüber nachdenken zu müssen, wäre die Person nacheinander ein Konsument in einem Markt, ein Staatsbürger und ein Anarcho-Kommunist in einer Geschenksökonomie gewesen. Die *Neue Ökonomie* des Netzes ist eine fortschrittliche Form sozialer Demokratie." (Barbrook 1998)

In der Literatur findet sich für eine solche Mischform bzw. Koexistenz häufig auch die Metapher der Allmende.[17] Aus der Geschichte der Allmende ist jedoch bekannt, dass diese soziale Einrichtung einer sozialen Für- und Vorsorge ein fragiles Gebilde und vielen Begehrlichkeiten ausgesetzt war. Bei aller intuitiven Stimmigkeit des Vergleichs sind hier aber auch differierende Momente zu benennen: Wenn ich mir mit einem anderen eine Weide teile, habe ich nur die Hälfte des Futters zur Verfügung. Wenn ich dagegen mein Wissen mit jemanden teile, habe ich deswegen nicht weniger, vielmehr die Chance auf ein Mehr. Hier wird mit Blick auf Open Source eine Ökologie des Geistes weiterzuentwickeln sein.[18]

Es dürfte deutlich geworden sein, dass die Antwort auf die Frage nach der Rückkehr der Utopie je nach gesellschaftspolitischer Verortung ganz unterschiedlich ausfallen dürfte. Während die von Barbrook attackierten Theorie-Jockeys[19] die Frage bejahen könnten, würden wir hier eine weniger emphatische und verhaltenere Auffassung vertreten. Danach wird über Open Source die Erinnerung (z. B. Allmende) an andere, nicht kapitalistische Lebensformen wachgehalten, gleichwohl wird die Zukunft wohl eher durch eine Koexistenz verschiedener Produktionsformen im oben angegebenen Sinne bestimmt sein. Vielleicht bedeutet dies nach dem Ende des Ost-West-Konfliktes sogar einen Utopieverlust.

17 Vgl. http://de.wikipedia.org/wiki/Allmende, http://de.wikipedia.org/wiki/Tragik_der_Allmende, http://de.wikipedia.org/wiki/Wissensallmende, Reckmann (2004).

18 Siehe hierfür Bateson (1994), Spinner (2000) „Karlsruher Ansatz der integrierten Wissensforschung" (KAW) mit den drei Hauptfeldern des Wissensarten-, Wissensordnungs- und Wissensverhalten-Projekts.

19 Nach Barbrook handelt es sich dabei um einen Amsterdamer Slang für Intellektuelle, die mit Philosophien *cut 'n' mix* betreiben, wie DJs in einem Club.

Literaturverzeichnis

Acker, K. und Hettich, F. (2001), 'Open Source – Non-Profit-Engagement oder Service Geschäft?'. Robert Goecke (Hrsg.) Arbeitsbericht der Veranstaltung „Teledienste – Trendanalyse und Bewertung" am Lehrstuhl für Allgemeine und Industrielle Betriebswirtschaftslehre der Technischen Universität München. http://www.segma.de/vorlesung00/opensource.pdf [1. Nov 2004].

Balke, F. (1998), *Gilles Deleuze*, Campus Einführungen, Campus, Frankfurt.

Barbrook, R. (1998), 'Die heiligen Narren. Deleuze, Guattari und die Hightech Geschenkökonomie', *Telepolis* . Aus dem Englischen von Barbara Pichler – Artikel-Nr. 6344 – http://www.telepolis.de/deutsch/special/med/6344/1.html [31. Okt 2004].

Barnes, B. (Hrsg.) (1972), *Sociology of Science: Selected readings*, Penguin.

Bateson, G. (1994), *Ökologie des Geistes: Anthropologische, psychologische biologische und epistemelogische Perspektiven*, number 571 *in* 'Suhrkamp Taschenbuch Wissenschaft', 5. Aufl., Suhrkamp Verlag, Frankfurt a. M.

Breuer, I., Leusch, P. und Mersch, D. (1996), *Welten im Kopf. Profile der Gegenwartsphilosophie. Band 3: England/USA*, Rotbuch Verlag, Berlin. Kurztext: http://www.momo-berlin.de/Mersch_Welten_3.html [14. Nov 2004].

Collins-Sussman, B., Fitzpatrick, B. W. und Pilat, C. M. (2004), *Version Control with Subversion*, O'Reilly, Sebastopol. http://svnbook.red-bean.com/svnbook/book.pdf [11. Jul 2004].

Diamond, D. und Torvalds, L. (2002), *Linus Torvalds: Just For Fun. Wie ein Freak die Computerwelt revolutionierte.*, Deutscher Taschenbuch Verlag, München.

Galtung, J. (1993), *Eurotopia. Die Zukunft eines Kontinents*, Promedia Verlag, Wien.

Görlich, C. F. und Humbert, L. (2003), Zur Rolle der Informatik im Kontext der mehrphasigen Lehrerbildung, *in* P. Hubwieser (Hrsg.), 'Informatik und Schule – Informatische Fachkonzepte im Unterricht. INFOS 2003 – 10. GI-Fachtagung 17. – 19. September 2003', München, S. 89–99. http://www.ham.nw.schule.de/pub/bscw.cgi/d44685/Informatik_Lehrerbildung_N.pdf [14. Nov 2004].

Hagstrom, W. O. (1965), Gift Giving as an Organisational Principle in Science, *in* 'The Scientific Community', Basic Books, S. 12–22.

Klafki, W. (1991*a*), Grundzüge eines neuen Allgemeinbildungskonzepts. Im Zentrum: Epochaltypische Schlüsselprobleme, *in* 'Neue Studien zur Bildungstheorie und Didaktik: Zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik', Beltz Verlag, Weinheim, Basel, S. 43 ff.

Künzli, A. (1986), *Mein und Dein. Zur Ideengeschichte der Eigentumsfeindschaft*, Bund Verlag, Köln. ISBN: 3-7663-0916-1.

Luthiger, B. (2004), Alles aus Spaß? Zur Motivation von Open-Source-Entwicklern, *in* B. Lutterbeck und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 93–106.

Lutterbeck, B. und Gehring, R. A. (Hrsg.) (2004), *Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell*, Lehmanns Media, Berlin. ISBN: 3-936427-78-X, http://www.Think-Ahead.org/ [19. Jun 2004].

Mannheim, K. (1928), Die Bedeutung der Konkurrenz im Gebiete des Geistigen, *in* 'Der Streit um die Wissenssoziologie. Erster Band: Die Entwicklung der deutschen Wissenssoziologie', S. 325 ff.

Mattern, F. (2002), 'Zur Evaluation der Informatik mittels bibliometrischer Analyse', *Informatik Spektrum* **25**(1), S. 22–32. Folien (Dagstuhl, März 2002) http://www.vs.inf.ethz.ch/publ/slides/bibliometroSlides.pdf [6. Nov 2004].

Meja, V. und Stehr, N. (Hrsg.) (1982), *Der Streit um die Wissenssoziologie. Erster Band: Die Entwicklung der deutschen Wissenssoziologie*, Suhrkamp, Frankfurt a. M.

Merton, R. K. (1972), Die Priorität bei wissenschaftlichen Entdeckungen. Ein Kapitel der Wissenschaftssoziologie, *in* P. Weingart (Hrsg.), 'Wissenschaftssoziologie', Vol. 1. Wissenschaftliche Entwicklung als sozialer Prozess, S. 121 ff.

Perlis, A. J. (1982), 'Epigrams on Programming', *SIGPLAN Notices* **17**(9), S. 7–13. http://www-pu.informatik.uni-tuebingen.de/users/klaeren/epigrams.html [7. Nov 2004].

Reckmann, H. (2004), 'Pädagogische und gesellschaftliche Potenziale freier Software am Beispiel von Linux'. http://fsub.schule.de/bildung/potenzial/potenzial.htm [15. Nov 2004].

Schröder, M. (2004), 'Projekt-Hosting für TEX-Entwickler', *Die TEXnische Komödie* **16**(1), S. 26–28.

Spinner, H. F. (2000), 'Karlsruher Ansatz der integrierten Wissensforschung (KAW)'. http://www.rz.uni-karlsruhe.de/~Helmut.Spinner/3/B/IV/index.html [21. Nov 2004].

Stock, W. G. (2000), Was ist eine Publikation? Zum Problem der Einheitenbildung in der Wissenschaftsforschung, *in* K. Fuchs-Kittowski, H. Laitko, H. Parthey und W. Umstätter (Hrsg.), 'Wissenschaftsforschung Jahrbuch 1998', S. 239–282. http://www.wissenschaftsforschung.de/JB98_239-282.pdf [7. Nov 2004].

Storer, N. W. (1972), Das soziale System der Wissenschaft, *in* P. Weingart (Hrsg.), 'Wissenschaftssoziologie', S. 60–81.

Tanenbaum, A. S. und Torvalds, L. B. (1992), Appendix A: The Tanenbaum-Torvalds Debate, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources. Voices from the Open Source Revolution', S. 221–251. http://www.oreilly.com/catalog/opensources/book/appa.html [7. Nov 2004].

Weingart, P. (Hrsg.) (1972), *Wissenschaftssoziologie*, Vol. 1. Wissenschaftliche Entwicklung als sozialer Prozess, Athenäum, Frankfurt a. M.

Weingart, P., Pansegrau, P. und Winterhager, M. (Hrsg.) (1998), *Die Bedeutung von Medien für die Reputation von Wissenschaftlern*, Arbeitsbericht zum Lehrforschungsprojekt, Universität – Fakultät für Soziologie, Bielefeld. http://www.uni-bielefeld.de/iwt/mw/lf/ [7. Nov 2004].

Infrastrukturen der Allmende – Open Source, Innovation und die Zukunft des Internets

BERND LUTTERBECK

Als Allmende organisierte Gebilde, wie zum Beispiel Straßen können in hohem Maße Innovationen erzeugen. Man kann inzwischen gut begründen, welche Eigenschaften für dieses Potential verantwortlich sind. In der Literatur und vielen politischen Debatten scheint man sich einig zu sein, dass auch das Internet eine Allmende oder ein *Commons* ist. Allerdings zeigt eine Durchsicht der Literatur, dass die Gleichsetzung oberflächlich ist. Weil man die innovationsbegründenden Eigenschaften des Netzes nicht kennt, ist die Rede vom „Internet Commons" noch mehr ein Schlagwort als die gehaltvolle Beschreibung eines Ortes. Der Beitrag schlägt vor, die noch offenen Fragen dadurch zu klären, dass man einen Umweg geht und das Internet als *Infrastruktur* betrachtet. Dadurch wird es möglich, an die langjährigen Debatten und die reichhaltige Literatur um klassische Infrastrukturen anzuknüpfen und eine Analogie herzustellen. Dies wird am Beispiel der Infrastruktur *See* verdeutlicht. Der Beitrag kommt zum Ergebnis, dass die Analogie möglich ist. Allerdings kann die Infrastruktur *Internet* nur verstanden werden, wenn man sie mit ingenieurswissenschaftlichen Konzepten über Kommunikation verbindet. Das Internet kann und muss als Allmende organisiert werden, will man das unbestrittene Innovationspotential des Internets erhalten. Da die Diskussion in Deutschland noch am Anfang steht, werden abschließend einige Maßnahmen vorgeschlagen, mit denen sich kurzfrig Verschlechterungen des Status Quo vermeiden ließen.

1. Einleitung

Irgendwie wird man bei dem deutschen Wort Allmende den Gedanken an braunäugige Kühe nicht los, die auf einer blühenden Alm glücklich vor sich hin grasen. Allmende ist auch in den einschlägigen Lexika zu allererst eine Gemeindewiese. Die deutsche Rede von der Allmende klingt deshalb ein wenig altfränkisch.

In der englischen Sprache hat das Wort *Commons* einen anderen Klang, nicht erst, seit Larry Lessig seine *Creative-Commons*-Initiative ins Leben gerufen hat.[1] Natürlich denkt man auch dort an das freie Land, das von guten Farmern genutzt und von habgierigen Schurken geraubt wird – Kampf um Land, Wasser und Vieh, ein Hollywood-Thema bis in unsere Tage. Hinzugekommen sind in jüngerer Zeit wissenschaftlich bahnbrechende Arbeiten über die Tragödie (Hardin 1968) oder Komödie (Rose 1986) der Allmende und das *Anticommons* (Heller 1998, Heller und Eisenberg 1998). Seit dem Manifest des Juristen Boyle (1997) ist die Diskussion um das *Commons* auch aus den Debatten um die gesellschaftliche Organisation des Internets und der Informationsgesellschaft als Ganze nicht mehr wegzudenken: „ . . . the Internet currently is managed as a commons" (Frischmann 2004, S. 85).

Es hat den Anschein, dass als Allmende organisierte Gebilde in hohem Maße Innovationen hervorbringen – jedenfalls gilt das für klassische Allmenden, wie zum Beispiel Straßen. Man ist sich sicher, dass Gleiches auch für das Internet gilt.[2] Da das Internet aber ungeplant entstanden ist, fehlt es an gesichertem Wissen. Wer immer das Netz in der Zukunft gestalten will oder muss, sollte wissen, welche Eigenschaften dieses Innovationspotential begründen. Man sollte wissen, wo Regulation hilft und wo sie schädlich ist.

In diesem Beitrag vertrete ich die These, dass man Antworten findet, wenn man das Netz oder allgemeiner die neu entstehende Struktur für Information, Wissen und Kommunikation als „Infrastruktur" betrachtet.[3] Die grundlegende und für das soziale und ökonomische Gedeihen der Volkswirtschaften entscheidende Frage ist: Inwieweit soll das Internet auch in der Zukunft als Allmende organisiert werden?

Das Internet ist nur durch Open Source groß geworden. Die Antwort auf die Frage scheint deshalb nahe zu liegen. Aber Vorsicht: Das Terrain ist glitschiger, als sich vielen schon erschlossen hat. Nicht zuletzt deshalb, weil jede ernst zu nehmende Diskussion um Allmenden, Geistiges Eigentum, Public Domain und damit auch Open Source *im Schatten* der Provokation von Garret Hardin stattfindet: der Tragödie der Allmende (Boyle 2003, S. 7).

---

1 Allmende ist eine Ressource, die gemeinsam genutzt und deren Zugriff offen für alle Nutzer ist – unbeschadet ihrer Identität oder des intendierten Gebrauchs (Lessig 2001, S. 19–20).

2 Ich betrachte hier das Internet als Modell des Ganzen. Zu diesem methodischen Ansatz vergleiche man schon mein frühes Thesen-Papier von 1997 „Ist das Internet ein Modell der Informationsgesellschaft?" (Lutterbeck 1997).

3 „Der Sammelbegriff Infrastruktur bezeichnet alle langlebigen Grundeinrichtungen personeller, materieller und institutioneller Art, die das Funktionieren einer arbeitsteiligen Volkswirtschaft garantieren" (Wikipedia).
Frischmann (2004, S. 37) entwickelt ein *demand-side model of infrastructure*, das der hier vertretenen Sicht sehr nahe kommt: „Infrastructure resources are resources that satisfy the following demandside criteria: (1) The resource is (or may be) consumed nonrivalrously (2) Social demand for the resource is driven primarily by downstream productive activity that requires the resource as an input (3) The resource is used as an input into a wide range of goods and services, including private goods, public goods and/or non-marketgoods."

2. Die Tragödie der Allmende ist aufhaltsam!

Betrachten wir einen See.[4] Einen See in einer schönen Umgebung, zum Beispiel mit Bergen im Hintergrund. Man könnte an Oberbayern oder Kärnten denken. Es ist Sommer, richtig schöner Sommer.

Offensichtlich hängt es von der Perspektive des Betrachters ab, wie man diesen See sieht. Man kann an Freizeit denken, an einen Ort der Erholung: Schwimmen, die Landschaft betrachten. Oder an Sport: Kanu fahren, Surfen, Angeln. Oder an Geld verdienen: Fischen und Restaurants, in denen man Fisch verzehren kann, Souvenirläden und Hotels. Man kann an Wasserkraft und Strom und Wasser für die Allgemeinheit denken, also den Aspekt der Vorsorgung der Allgemeinheit hervorheben. Oder die seltenen Vögel, die einen bestimmten Teil des Sees als Brut- und Ruheraum benutzen.

Man kann natürlich auch schwärmen, dann ist für irgendwelche Nützlichkeitserwägungen kein Platz.

Keine Frage: Der See ist ein komplexes Gebilde. Je nach Standpunkt macht er eine Vielzahl von Nutzungen möglich oder verbietet sie. Dieser See ist eine Infrastruktur, wenn auch keine traditionelle.[5]

Gehen wir jetzt in eine andere Welt, die Welt der Technik, des Internets, der festen und mobilen Netze. Was betrachten wir jetzt?

In vielen politischen Debatten ist diese Frage eigentlich kein Streitpunkt mehr. Zu Unrecht: Denn eine Durchsicht der Literatur ergibt, dass wir eigentlich noch nichts Genaues wissen. Normativ scheint man sich einig zu sein. Dabei können wir den Ort, auf den sich unsere Wertungen beziehen, noch nicht einmal sprachlich präzis benennen. Die Unsicherheit ist groß, wie sich bei einem schon flüchtigen Blick in die Literatur der letzten Jahre ergibt:

- „digitally networked environment" (Benkler 2000)
- „interrelated resource facilities" (Bernbom 2000)
- „Internet commons" (Bernbom 2000)
- „open information environment" (Samuelson 2001)
- „information infrastructure" (Networked Readyness Index)[6]
- „core common infrastructure" (Benkler 2003)
- „information commons" (Cahir 2003)
- „infrastructure commons" (Frischmann 2004)
- „digital communications platform" (Cooper 2004)

4 Frischmann (2004) und Hess und Ostrom (2003), die dieses Beispiel an mehreren Stellen benutzen, haben mich zur Wahl dieses Beispiels animiert. Man müsste im Text ein wunderschönes Bild von einem See abdrucken, natürlich bunt. Die Leserin möge stattdessen ihre Phantasie walten lassen.

5 Frischmann (2004, S. 6) unterscheidet vier traditionelle Infrastrukturen: (1) transportation systems, (2) communications systems, (3) governance systems und (4) basic public services and facilities.

6 „In previous work we defined Networked Readiness as 'the degree to which a community is *prepared* to participate in the Networked World.' In this report, we expand that definition to include a community's *potential* to participate in the Networked World in the future" (Kirkman et al. 2002, S. 11).

– „electronic communications network“ (Whitt 2004)
– „the nation's communications infrastructure“ (Wu 2004)
– „innovation commons“ (Wu 2004)
– „elektronische Kommunikationsinfrastruktur“ (Rahmenrichtlinie der Europäischen Union von 2002 für Kommunikationsnetze)[7]

Offensichtlich ist, dass manche Autoren eine Analogie zur Umwelt versuchen und so Umwelt als normativen Input für ihre Sicht dieser neuen Netze benutzen. Die Analogie als solche ist nicht das Problem, sondern der Verzicht auf eine empirisch gehaltvolle Beschreibung des Ortes, den man organisieren will. Das wird zum Beispiel deutlich in der zitierten Richtlinie der Europäischen Union über Kommunikationsnetze. Das Wort „Kommunikationsinfrastruktur“ wird mehrfach benutzt, aber nie definiert. Aus dem Kontext ergibt sich, dass dieser Begriff so ziemlich alles erfasst, was einem in diesem Zusammenhang einfällt: Netze, mobile wie satellitengestützte, Dienste, Einrichtungen, Zugangssysteme, nationale und internationale Märkte, Fernsehgeräte und die APIs, die Schnittstellen für Anwendungsprogramme, die Verbraucher, die Nutzer, die Behörden.

Wenn schon führende Wissenschaftler nicht wissen, was sie eigentlich genau untersuchen, muss man es einem Europäischen Gesetzgeber gewissermaßen nachsehen, wenn er nicht so genau weiß, was er reguliert. Allerdings steht zu befürchten, dass das allseitige Nicht-Wissen sich gegenseitig verstärkt und damit das bedroht, was eigentlich geschützt werden soll: das Innovationspotential des Netzes.

Im Recht gibt es einen erprobten Weg, um solche unübersichtlichen Situationen zu umschiffen: Man geht vom Sicheren zum Unsicheren aus und versucht die Analogie. Man fragt: Warum können bestimmte klassische Ressourcen erfolgreich als Allmende organisiert werden, andere aber nicht? Warum lohnt es sich überhaupt, etwas als Allmende zu konzeptualisieren? Erst wenn man diese Frage geklärt hat, ist Raum für die weitere: Warum sollten auch andere, nicht klassische, völlig neuartige Ressourcen wie das Internet als Allmende organisiert werden?

Dieser Gedanke bringt mich zurück zu meinem Eingangsbeispiel: dem See. Man kann versuchen, die verschiedenen Aktivitäten, Zugriffswünsche und Restriktionen zu typisieren und den Typen ihr jeweiliges Eigentumsregime zuordnen. Denn es macht einen Unterschied, ob ich nur schwimmen will. Dann brauche ich nur ungehinderten Zugang. Oder ob ich angeln oder Wasser entnehmen will. Noch mal anders ist es, wenn ich über das Regime der Zuflüsse oder den Vogelschutz entscheiden muss. Hier muss man andere am Zugang hindern, ggf. durch aktive Maßnahmen – und das unabhängig davon, wer Eigentümer ist.

In einem ersten Schritt kann man die Rechte an bestimmten Eigenschaften des Sees definieren. Man spricht hier auch von dem Bündel der Eigentumsrechte.

7 Richtlinie 2002/21/EG des Europäischen Parlaments und des Rates vom 7.März 2002 über einen gemeinsamen Rechtsrahmen für elektronische Kommunikationsnetze und -dienste (Rahmenrichtlinie), Amtsblatt der Europäischen Gemeinschaften L 108/33 vom 24. 4. 2002.

Recht	Beschreibung	Anwendung
Zugriff *Access*	Recht, eine definierte Gegend zu betreten; Genuss von Vorteilen	Kanu fahren; schwimmen; die Natur genießen
Entnahme *Extraction*	Recht, Ressourceneinheiten zu entnehmen	fischen; Recht, Wasser abzuleiten
Management	Recht, Benutzungsrechte zu definieren	Verbot für Motorboote; Vogelschutz
Ausschluss Dritter *Exclusion*	Recht, Zugriffsrechte zu definieren	
Veräußerung *Alienation*	Recht, Eigentum an Dritte zu übertragen	

Tabelle 1: Das Bündel von Eigentumsrechten, Quelle: Hess und Ostrom (2003, S. 124)

Das Veräußerungsrecht, *the right to sell or lease*, ist in der klassischen Theorie das definierende Etwas für Eigentum. Fehlt dieses Veräußerungsrecht, so sagt die herrschende Doktrin, ist Ineffizienz der verbleibenden Rechte die zwangsläufige Folge. Sonst sei die Tragödie der Allmende unvermeidlich: Koordination misslingt systematisch. Jeder bedient sich nach seinem Eigennutz und am Ende ist alles dahin.

In der einfachen Welt dieser Eigentumsdoktrin gibt es nur zwei Wege, diese Koordinationsprobleme zu lösen. Entweder ist eine Ressource Privateigentum oder sie ist öffentliches Eigentum. Schon ein erstes Durchspielen dieser Alternative wirft mehr Fragen auf als Antworten. Ein privater Eigentümer könnte natürlich Dritten jedes Recht vorenthalten. Das wäre bei der Attraktivität dieser Ressource volkswirtschaftliche Verschwendung. Denn all die vielfältigen Nutzungen des Sees könnten ja nicht entstehen. Auch ein privater Eigentümer müsste aber zum Beispiel Umweltbelange erfüllen. Ein öffentlicher Eigentümer bräuchte eine Art Seebehörde, um das öffentliche Interesse gerecht zu verwalten. Eine offene Frage ist es, wie dieser Eigentümer das Nutzungsinteresse um den See herum befriedigen würde.

Diese klassische Doktrin führt also in Probleme – einerlei, ob der Staat oder etwa eine Adelsfamilie Eigentümer des Sees ist.

Seit Hardin 1968 die Kontroverse um die Tragödie der Allmende ausgelöst hat, arbeitet die Wissenschaft mit Nachdruck daran, derartige Dilemmata zu vermeiden. Der Ertrag ist beeindruckend, die wichtigsten Ergebnisse lassen sich so zusammenfassen:

1. Es gibt mehr als zwei Typen von Gütern.

2. Man muss den Typ eines Gutes und das Eigentumsregime voneinander unterscheiden.

Ausschliessbarkeit von der Nutzung	Erschöpfbarkeit der Ressource	
	Gering	Hoch
Schwierig	Öffentliche Güter Verteidigung Sonnenuntergang Öffentliches Wissen *Nicht verstopfte* *Straßen*	Allmenderessourcen Fische im Ozean Umwelt Bibliotheken *Verstopfte* *Straßen*
Leicht	Clubgüter Feuerwehr Segelklub *Nicht verstopfte* *Mautstraßen*	Private Güter Eiscremekugeln Personalcomputer *Verstopfte* *Mautstraße*

Tabelle 2: Typen von Gütern, Quelle: Mankiw (2004, S. 224), Hess und Ostrom (2003, S. 120)

3. Man muss das Ressourcensystem und die Elemente der Ressource voneinander unterscheiden: Fisch und See sind nicht das Gleiche. Der Fisch kann einem Fischer gehören, der See aber der Allgemeinheit.

4. Eine Allmende kann das effizientere Managementsystem sein – falls bestimmte Bedingungen erfüllt sind.

Die Tabelle 2 fasst diese Gedanken nochmals zusammen:

Die entscheidende Aussage der Tabelle 2[8] ist die empirische Feststellung, dass etwas je nach Zustand in alle Typen fallen und dass der Typ des Gutes sich je nach den Gegebenheiten verwandeln kann. Allzu simple Eigentumsvorstellungen scheitern an der Wirklichkeit.

Hierzu wieder der See als Beispiel. Ein typischer Strandausschnitt könnte wie in Abbildung 1 aussehen. Ein bestimmter Teil ist eine Allmende, die als für die Öffentlichkeit zugänglicher Strandausschnitt genutzt wird. Daneben, durchaus auf privatem Grund und durch einen Zaun abgegrenzt, könnte eine Surfschule ihre Leistungen anbieten. Daneben könnte man sich eine Streuobstwiese vorstellen, die das Areal einer öffentlichen Wassergewinnung abgrenzt – ein Stück öffentliches Eigentum. Über diesen Nutzungen finden weitere Nutzungen statt, Nutzungen, die teilhaben am Zugriff zur Ressource See.

8 Bei Hess und Ostrom (2003, S. 120) und vielen anderen erscheint *Clubgut* statt natürliche Monopole in der Abbildung. „Als Clubgut werden Güter bezeichnet, bei denen Ausschließbarkeit im Konsum oder von der Nutzung möglich ist und eine zumindest partielle Rivalität im Konsum vorliegt. Beispiele für Clubgüter sind Tennis- oder Golfclubs. Bei steigender Mitgliederzahl tritt bei derartigen Clubs durch Kapazitätsgrenzen Rivalität im Konsum auf, wodurch ab einer bestimmten Grenze ein Ausschluss weiterer Mitglieder vorgenommen wird“ (Wikipedia).

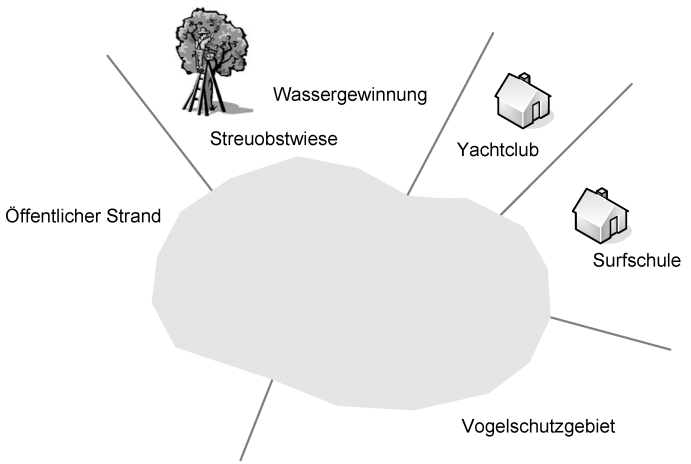


Abbildung 1: Der See konkret

Über diese Zusammenhänge besteht unter Ökonomen zumindest kein prinzipieller Streit. Damit sind aber nur Extremvorstellungen über das Eigentum abgewehrt und ein Analyseinstrumentarium bereitgestellt. Noch unbeantwortet ist damit eine weitere Frage: Warum soll eine Infrastruktur so und nicht anders organisiert werden? Welches normative Prinzip organisiert die Allmende?

> „And generally speaking, practises that enhance the sociability of the practitioners have greater returns with greater scale: one cannot get too much of them. If the inherently public property doctrines protect not just commerce, but socialising activities in general, we should expect that other socializing activities might also give rise to inherently public property insofar those activities require certain specific locations“ (Rose 1986, S. 777).

Infrastrukturen wie zum Beispiel Straßen sind Ressourcen, die ihren Zweck nicht in sich selber tragen, sondern etwas ermöglichen sollen. Meist sollen sie den Kommerz fördern. Sie haben die Fähigkeit, unseren Wohlstand zu mehren und auf der Nachfrageseite positive Externalitäten[9] zu erzeugen. Deshalb, also aus einem ökonomischen Grund, seien sie besonders gute Kandidaten für die Allmende. Nicht nur deshalb, sondern vor allem auch, weil sie besonders gut die soziale Kohärenz der Menschen fördern.

9 „An externality arises when a person engages in an activity that influences the well-being of a bystander and yet neither pays or receives any compensation for that effect. If the income of bystander is adverse, it is called a *negative externality*.; if it is benificial it is called *positive externality*“ (Mankiw 2004, S. 204).

Wenn das der Fall ist, müssten sich die Vorzüge von Allmende-Eigentum auch auf anderen Feldern gesellschaftlicher Praxis bewähren.[10]

Ein solches gesellschaftliches Feld könnte die Benutzung der „Kommunikationsinfrastruktur" unserer Gesellschaften sein.

Die Menschen bringen also durch ihre soziale Praxis Infrastrukturen gewissermaßen *zum Schwingen*: Aus dem Stück wird eine Komödie. Text und Handlung werden sich unterscheiden, die Essenz bleibt immer die gleiche: Innovation. Das Ergebnis deckt sich mit den Behauptungen der Netzwerkökonomie. Danach ist Innovation die Übernahme einer neuen Praxis durch eine Gemeinschaft. Innovation ist also immer eine soziale Transformation (Denning 2004). „Jede Innovation ist soziale Innovation" (Tuomi 2003, S. 5).[11] Im klassischen Innovationsmodell denkt man *upstream*. Im Allmende-Modell denkt man *downstream* – von dem Ort, an dem soziale Kommunikation sich ereignet.[12] Die nachgelagerten Aktivitäten geben der Infrastruktur *See* Gestalt.

3. Die Komödie der Allmende – gelayert

Das Ergebnis dieser Diskussion ist außerordentlich ermutigend: Uns erwartet die Komödie, nicht die Tragödie der Allmende – vorausgesetzt, es gelingt, identifizierte Prinzipien auch in dem neuen, durch das Internet geformten Bereich nachzuweisen. James Boyle hat die Notwendigkeit, eine Analogie zu suchen, schon 1997 überzeugend begründet (Boyle 1997). Allerdings waren sein Manifest und die vielen Konzepte, die es übernahmen, ausschließlich normativ verankert und noch ungeeignet, Gestaltungshinweise für das Neue zu geben.

Wo musste man suchen? Wo ist der empirische Ort, über den sich die Analogie begründen ließ?

10 Die von Carol Rose untersuchten Allmenden haben also drei wesentliche Elemente: einen empirischen Ort, einen Zweck und einen normativen Rahmen.

11 Das deckt sich mit der Auffassung von Schumpeter (1997, S. 110 ff.), wonach Innovation die *Durchsetzung* von Etwas durch den schöpferischen Unternehmer ist. Das unterscheidet Innovation von Invention, der bloßen Erzeugung von Ideen und technischen Artefakten.

12 Hier übernimmt die Ökonomie Worte aus der Informatik, die Begrifflichkeit ist aber intuitiv nicht eben leicht verständlich. Jeder normale Nutzer kennt das Wort *download*, das auf der Unterscheidung von *upstream* und *downstream* aufsetzt. Es bezeichnet den Befehl für zum Subjekt – im Regelfall also zum Nutzer – gerichtete Datenströme, das Wort *upload* für vom Subjekt abgehende. Jeder normale Nutzer etwa eines DSL-Anschlusses weiß auch, dass die *upstream*- und *downstream*-Prozesse in der Regel asymmetrisch sind – um Größenordnungen. Jemand, der die *upstream*-Prozesse etwa über die Bandbreite kontrolliert bzw. bewusst verlangsamt, kann dadurch die Möglichkeiten des Nutzers zum Senden von Informationen (Webseiten, Sprache usw.) bewusst einschränken und ihn so – überzeichnet – zum „Informationskonsumenten" ohne eigene Gestaltungsmöglichkeiten „degradieren". Dadurch würde das Innovationsgeschehen auf der Seite der Nutzer zumindest verlangsamt. Ähnlich wie in der ökonomischen Begrifflichkeit geht es hier also nicht nur um ein technisches Problem, sondern auch um Kategorien von *unten* und *oben* – *unten* stehen die Wirtschaftssubjekte bzw. Nutzer, die von *oben* zum Beispiel reguliert werden. Die Worte *upstream* und *downstream* erfassen also technische Sachverhalte ebenso wie gesellschaftliche Hierarchien und Ungleichgewichte, etwa das Problem der Informations-Asymmetrie. Jeder weiß, dass es *unten* und *oben* gibt. Fraglich ist, ob das so bleiben soll.

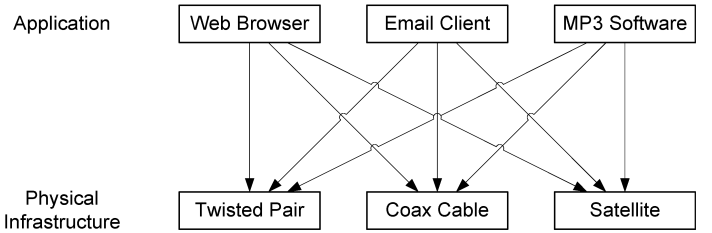


Abbildung 2: Infrastruktur – ohne Layer, Quelle: Whitt (2004, S. 361)

Carol Rose hatte es noch vergleichsweise einfach. Sie konnte sich auf eine teilweise Jahrhunderte währende Diskussion um Straßen, Seen, Strände, Weiden und Watten usw. berufen. Immer fand sie einen Ort vor, über den sich der ökonomische Nutzen der Allmenden begründen ließ. Die Diskussion um den sog. *Cyberspace* war ganz neu und gerade einmal einige Jahre alt und die massiven ökonomischen und politischen Interessen geben kaum Zeit, in Ruhe über die Dinge nachzudenken. Die Antwort der Wissenschaft ist vielleicht überraschend:

1. Es ist schwierig den Ort zu finden. Man kann aber den Zweck finden, der Orte definiert, das reicht.

2. Man muss bei den Ingenieurswissenschaften nach Lösungen suchen.

Von Carol Rose haben wir gelernt, dass ein Ort seine Bedeutung durch die kommunikative Praxis der Gesellschaft erhält. Dadurch ist die Gesellschaft in der Lage, ökonomisch Reichtum zu erzeugen.

Von den Ingenieurswissenschaften muss man lernen, dass Kommunikation umso besser gelingt, je präziser sie gelayert ist.

Ein Ort ist dann definiert durch seine Layer oder zu Deutsch „Schichten". Jeder Layer ist definiert durch bestimmte Regeln, etwa für Formate und Definitionen der Schnittstellen zwischen Layern. Würde man Kommunikation nicht so aufteilen, dann müsste jede einzelne Änderung in einem Teil zu Änderungen im Gesamtsystem führen. Das wäre ökonomisch ineffizient und vor allem fehleranfällig. Solche Layer erleichtern zum Beispiel das Management von Netzwerken. Layer sind also notwendige, wenn auch nicht hinreichende Bedingungen gelingender Kommunikation. Sie konstituieren die Infrastruktur des *Commons*.[13] Dies lässt sich am Beispiel illustrieren, wie es in Abbildung 2 und 3 zu sehen ist:

Die vielen einzelnen Aspekte von Kommunikation lassen sich zu Modellen zusammenfassen. Allen Modellen gemeinsam ist, dass sie Kommunikation hierarchisch in horizontale Layer auflösen und in den Layern zusammengehörende Funktionalitäten

13 In einem Vortrag vom März 2004 in Stanford hat Frischmann die These aufgestellt: „If infrastructure then commons." Die These ist natürlich ein wenig simpel, sei aber ein gutes *organizing principle*, unter http://cyberlaw.stanford.edu/events/archives/brett_m_frischmann.shtml.

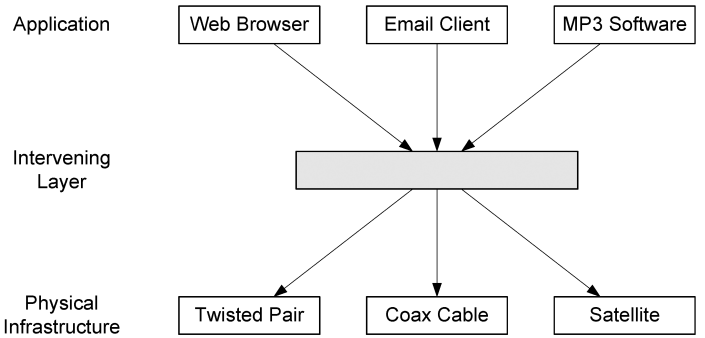


Abbildung 3: Infrastruktur – mit Layer, Quelle: Whitt (2004, S. 361)

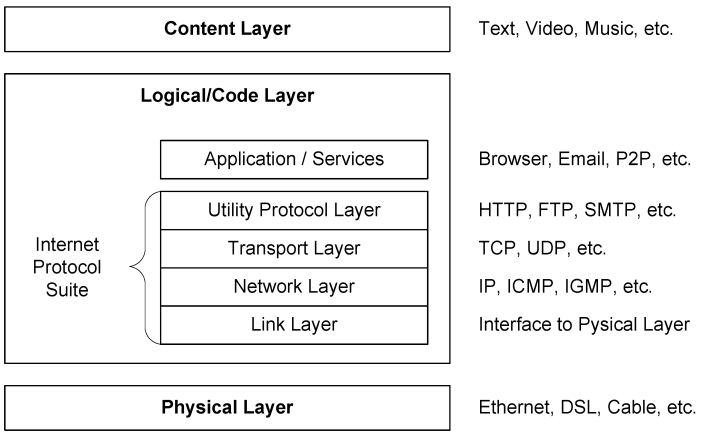


Abbildung 4: Das Layer-Modell von Benkler, Quelle: Benkler (2000), Visualisierung von Whitt (2004, S. 365)

bündeln. Yochai Benkler, der in Yale Recht lehrt, hat das gegenwärtig gebräuchlichste Modell (siehe Abbildung 4) entwickelt:[14] Das Entscheidende dieser horizontalen Layer ist, dass sie funktionale Komponenten eines End-to-End Systems sind. Jeder Layer operiert nach spezifischen Bedingungen und eigenen Regeln. In einem nächsten Schritt könnte man die normativen Vorschriften bestimmen, die für die einzelnen Schichten gelten sollen. Zum Beispiel könnte man Verbote aufstellen, einzelne Layer zu überschreiten.[15] Wohlgemerkt, dies sind normative Prinzipien, die ihr Fundament in den Ingenieurswissenschaften haben.

14 Benkler ist für Lessig (2001, S. 23) „perhaps the best communication theorist of our generation".

15 Rechtsprofessor L. Solum und der Unternehmensberater M. Chung haben das bisher umfangreichste Konzept vorgelegt, das das Layer-Modell normativ umsetzt (Solum und Chung 2003).

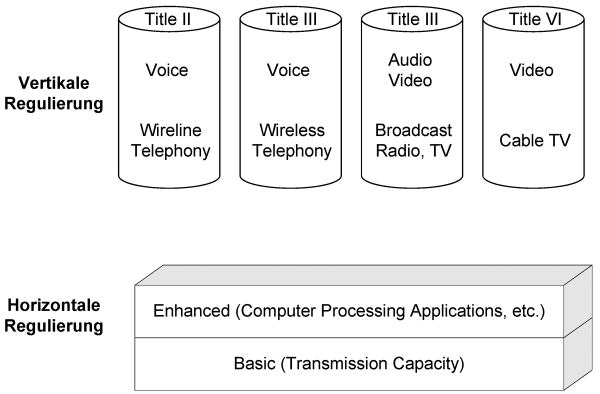


Abbildung 5: Vom Silomodell zum Schichtenmodell, Quelle: Whitt (2004, S. 356)

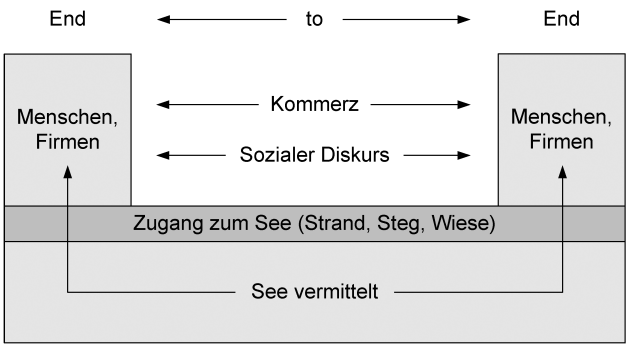


Abbildung 6: Der See gelayert

Ich strapaziere ein weiteres Mal die Analogie zu einem See. Die Fische und die Fischerei sind etwas anderes als Umweltauflagen oder Rechte der Menschen, den See zum Schwimmen zu benutzen. Bei der Kommunikation ist es im Prinzip nicht anders: Ich darf etwa den Inhalt eines Telefongesprächs nicht mithören, darf aber sehr wohl die entstehenden Gebühren kontrollieren. Und noch mal eine Frage ist es, wer Leitungen und Satelliten zur Übertragung besitzt.

Die Regulierungskonzepte haben diesen Zwang, Kommunikation horizontal zu layern, schon umgesetzt – in den USA ebenso wie in der Europäischen Union. Das Konzept vertikaler Regulierung, anschaulich Silo-Modell genannt, gehört der Vergangenheit an. Man muss also die horizontalen Schichten der Layer vom vertikalen Aufbau der Kommunikation unterscheiden.

Mit diesem Wissen komme ich ein weiteres Mal zu dem See zurück. Die Abbildung 6 zeigt die Abstraktion eines Sees, mit seinen wesentlichen Funktionalitäten. Die sozialen

Akteure kommunizieren End-to-End. Die soziale und kommerzielle Kommunikation geht aber gewissermaßen durch den See hindurch und hat die Struktur eines „U". Denn ohne den See, ohne die konkrete Infrastruktur bestünde ja kein Anlass zu kommunizieren. Die Kommunikation realisiert sich durch die Schnittstelle zum See: den Zutritt. Das Ergebnis ist eine Infrastruktur, die dazu beiträgt, die soziale Wohlfahrt zu mehren.

Nicht anders ist es bei netzgestützter Kommunikation: Die Akteure kommunizieren „U-förmig". Wann dürfen wir in diesem Fall auf eine Komödie der Allmende hoffen? Eine Antwort muss offensichtlich ingenieurswissenschaftliche Einsichten und ökonomische Prinzipien in sich vereinen. Tim Wu, ein gelegentlicher Mitautor von Lawrence Lessig, hat die Summe der möglichen Antworten vor kurzem so zusammengefasst. Das *Commons* gründet sich nach Wu (2004) auf drei Prinzipien:

- Infrastruktur-Prinzip
    - Quelle für Externalitäten
    - Wert ist indirekt; Spillover
- Neutralitätsprinzip
    - Keine Diskriminierung zwischen Inhalten, Nutzern und Nutzungen
- End-to-End Prinzip
    - Technisches Designprinzip für Netzwerkarchitekturen
    - *e2e* steht für eine Theorie der Innovation

Innovation ist ein evolutionärer Prozess. Informationstechnische Infrastrukturen sind, in einem schrecklichen deutschen Wort, „Ermöglichungsstrukturen", sie tragen also ihren Zweck nicht in sich selbst. Die Basis dieser Struktur, das Internet, ist eine sog. „General Purpose Technology", eine völlig unspezifische Technologie wie die Dampfmaschine, die Straßen, die Elektrizität, das Telefon. Erst Erfindungen vieler Einzelner – ich benutze das Wort Erfindung bewusst – geben dieser Struktur Gestalt.[16] Die Nachfrage nach diesem Gut wird getrieben durch nachgelagerte Produzenten, die die Ressource – dort der See, hier das Internet – als Grundlage für ihre ökonomischen Aktivitäten benötigen. Wieder der See als Beispiel: Ein entsprechend großer und schöner See ist attraktiv für viele kleine und vielleicht einige große Unternehmer. Sie können Hotels bauen, Boutiquen aufmachen, Naturkurse für das reifere Publikum anbieten. Sie produzieren darauf Güter. Die Alternative wäre, dass der Staat oder ein privater Eigentümer gewissermaßen an der Angebotsschraube dreht.

Der gleiche Zweck wird im Falle des Internets durch die Applikationen und die Nutzer, die sich ihrer bedienen, realisiert. Dieser Zweck muss durch eine technische Architektur und Prinzipien, die sie leiten, umgesetzt werden. Dies ist das Neutralitätsprinzip. Keine Diskriminierung von Nutzern und Nutzungen – in Szene gesetzt durch

16 Siehe zum Ganzen das Kapitel 11 der Arbeit von van Schewick (2004, S. 287 ff.) „Social Benefits of Different Economic Environments for Application-Level Innovation".

Abbildung 7: Der See als Infrastruktur

das End-to-End Prinzip. Der Code, die Software der horizontalen Layer definieren das End-to-End Kommunikationssystem. Die Architektur des Internets und seine frühen Erbauer, die Mathematiker und Ingenieure waren, behaupten, dass dieses Design eine größere Bandbreite von Innovationen liefert als andere Architekturen.

Das End-to-End Prinzip ist also eine ingenieurwissenschaftliche Innovationstheorie, eine Theorie, die die Notwendigkeit einer nachfrageorientierten Ökonomie betont. Wenn nicht alles täuscht, bestätigen empirische Untersuchungen im Bereich ökonomischer Innovationsforschung diese evolutionäre Sicht von Innovationsprozessen (Osterloh und Rota 2004, Osterloh et al. 2004).

Diese Theorie besagt also zusammengefasst: Innovationen sind umso häufiger und umso wahrscheinlicher, je mehr sich das System von den Anwendungen her entwickeln kann. Dies ist allein bei einer End-to-End Architektur der Fall.

4. Alles wird gut – hoffentlich!

Es siegt also die Komödie. Alles wird gut – theoretisch. Praktisch sind die Hürden hoch, hoffentlich nicht unüberwindlich hoch. Es muss schnell etwas passieren, weil die Bedrohungen des evolutionären Prinzips der Innovation nicht von der Hand zu weisen sind.

Kurzfristig sehe ich zwei Möglichkeiten, die Tagesordnung der Politik zu verändern. Man muss erstens eine normative Diskussion führen und darauf hoffen, dass die gemeinsamen Kenntnisse des zu regelnden Bereichs größer werden. Hier hat der ehemalige Chairman der amerikanischen Regulierungsbehörde FCC vor einiger Zeit einen Vorschlag gemacht, der nicht unterschritten werden sollte. Powell (2004) hat vier Internet-Freiheiten postuliert:

- Freiheit, auf Inhalte zuzugreifen
- Freiheit, die Anwendung der Wahl zu fahren
- Freiheit, Geräte der eigenen Wahl zu benutzen
- Freiheit, „to obtain service plan information“

Durch den offensichtlichen Erfolg von Offener Software sollte es möglich sein, Bündnisgenossen für die Durchsetzung dieser Freiheiten zu gewinnen.

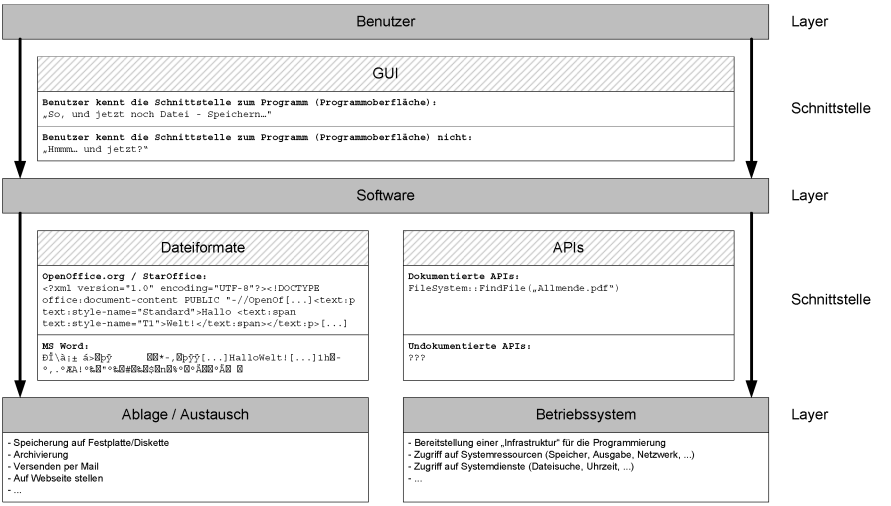


Abbildung 8: Schnittstellen

Wir müssten zweitens schnellstmöglich die End-to-End Diskussion führen. Die Schwierigkeit liegt aber darin, dass die deutsche Wissenschaft und die deutsche Politik das Problem noch kaum entdeckt haben. Dies ist ein großer Unterschied zu den USA, wo alle Beteiligten seit vielen Jahren um das technische Problem wissen. Es wäre daher nicht aussichtsreich, die dortigen Problemlösungen eins zu eins auf Deutschland zu übertragen. Eine solche Diskussion braucht Zeit, die politischen Lösungen müssen aber schon jetzt gefunden werden.

Was aber passiert, wenn diese technisch-ökonomische Perspektive noch nicht einmal als solche erkannt wird, lässt sich exemplarisch am Schicksal der Softwarepatentrichtlinie der Europäischen Union aufzeigen: Wie auch immer man das End-to-End Prinzip genau präzisieren muss, in jedem Fall sind die jeweiligen Layer auch durch ihre Schnittstellen zu den je angrenzenden definiert. Schnittstellen sind ein Teil der kommunikativen Infrastruktur und unerlässlicher Teil horizontaler Layer. Sind die Schnittstellen nicht offen, ist das Prinzip der Neutralität verletzt. Deshalb ist es für ein evolutionäres Prinzip zentral, dass die Schnittstellen jeweiliger Layer nicht patentiert werden dürfen oder dass die Verletzung von Patenten keine Folgen nach sich zieht. Der den europäischen Gremien vorliegende Vorschlag der Richtlinie enthielt bis zur Sitzung des Wettbewerbsrates am 24. Mai 2004 einen entsprechenden Vorschlag. Nach der aktuellen Fassung der Patent-Richtlinie der Europäischen Union sind derartige Patente nicht ausgeschlossen.[17] Dies widerspricht der durch das End-to-End Prinzip

17 Im Januar 2005 war noch offen, welchen Wortlaut der Rat der EU beschließen wird. Siehe hierzu KOM(2002) 92 – C5-0082/2002 – 2002/0047(COD), 24. September 2003 (http://www.computerundrecht.de/docs/legislative_entschliessung_des_europaeischen_parlaments_vom_24_9_2003.pdf) und RAT DER EUROPÄISCHEN UNION Brüssel, den 24. Mai 2004 (27. 05)

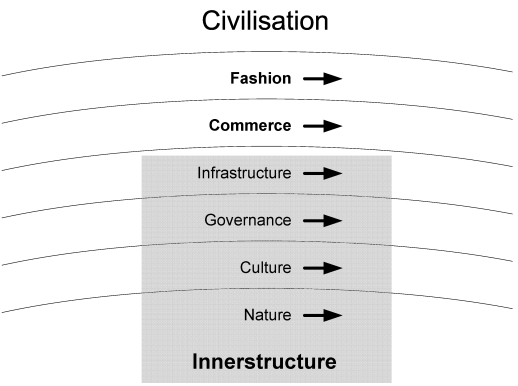


Abbildung 9: Infrastructure und Innerstructure, nach Searls (2002)

errichteten Architektur diametral.

Abbildung 8 fasst die wesentlichen Funktionalitäten von offenen und proprietären Schnittstellen zusammen. In einem Fall lässt sich das Geschehen nachvollziehen. Im anderen Fall ist man auf das Wohlwollen des Unternehmens angewiesen.

Da diese Schnittstellen im Prinzip bekannt sind, also keine aufwändige neue intellektuelle Aktivität angestoßen werden muss, müsste es möglich sein, die praktischen politischen Diskussionen der nächsten Zeit auf die Allmende der Schnittstellen zu konzentrieren. Man kann vermuten, dass die Fokussierung der Debatte auf diese technologische Kernfrage auch Erträge für die andere, viel größere Debatte über Infrastrukturen der Allmende abwirft. Diese Debatte gehört wohl eher in die Akademien und Seminare der Universitäten. Dabei könnte ein gedankliches Konzept von Doc Searls nützlich sein (Searls 2002). Searls, ein bekannter Open-Source-Entwickler und Herausgeber des Linux Journals, versucht die Welten des Normativen, Kulturellen und Zivilgesellschaftlichen mit den Ingenieurwissenschaften gedanklich zu vereinen: Er bewegt sich in dem durch Boyle vorgezeichneten normativen Rahmen, wendet ihn aber auf das Layer-Konzept an. Die hier in Ansätzen umrissene Infrastruktur der Allmende sei nur die Oberfläche von etwas viel Grundsätzlicherem. Darunter läge noch eine von ihm so genannte *innerstructure*: „Wie das Herz der Erde ist sie etwas, dass niemand besitzen und jeder nutzen kann. Wie die fruchtbare Oberfläche der Erde bezieht sie ihren Reichtum durch das Leben, das sie unterstützt. Sie schließt alle Layer ein – von der Infrastruktur bis hinunter zur Natur" (Searls 2002).

Im Ergebnis verweigert uns also die Politik noch den Zutritt in eine Komödie der Allmende. Trotzdem sollten wir uns die gute Laune nicht verderben lassen. Irgendwann wird sie begreifen müssen, dass man eine fortgeschrittene Gesellschaft nicht ohne Innovation voranbringen kann. Sie wird das primitive Trichtermodell der Innovation aufgeben müssen: Danach muss sie nur Geld, Patente und andere Ressourcen

(OR. en) Interinstitutionelles Dossier: 2002/0047 (COD) 9713/04 PI 46 CODEC 752.

in den Trichter hineinkippen und schon kommt Innovation heraus. Es ist genau umgekehrt: Die Nachfrager, also die Bürger und Unternehmer, benutzen die Infrastruktur für ihre Innovation. Sie halten das Rezept in der Hand, das den Reichtum erzeugt. *Downstream*, nicht *upstream* entsteht Neues.

Was macht eigentlich Carol Rose so sicher, dass wir am Ende doch in einer Komödie landen werden? Das *Commons* oder die Allmende hat sich über die Jahrhunderte gegen alle Widerstände durchgesetzt, weil es einen Ort gesellschaftlicher Praxis hat entstehen lassen. An und durch diesen Ort erzeugen Menschen gesellschaftlichen Reichtum. Diese doppelte Bedeutung der menschlichen Aktivität hat sich im Wort *Commerce* in der englischen Sprache noch erhalten.[18] Kommerz ist natürlich eine Aktivität, die eine Infrastruktur benutzt zum Zwecke des Kaufens und Verkaufens. Aber es ist auch die Aktivität des gesellschaftlichen, zivilisierten Umgangs miteinander. Eine Infrastruktur wird genutzt, um Ideen, Meinungen und Vorurteile auszutauschen. Indem Menschen sich so verhalten, erzeugen sie nachgelagerte Aktivitäten oder *downstream applications*. Es ist also das Grundprinzip der Zivilgesellschaft, das Allmenden groß gemacht hat. Das kann man für die alten Allmenden beweisen. Nichts spricht dafür, dass es im Falle des Internets anders ist.

Literaturverzeichnis

Benkler, Y. (2000), 'From Consumers to Users: Shifting the Deeper Structures of Regulation Toward Sustainable Commons and User Access', *Federal Communications Law Journal* **52**(3), S. 561–579.

Benkler, Y. (2003), 'The Political Economy of Commons', *European Journal for the Informatics Professional* **4**(3), S. 6 ff.

Bernbom, G. (2000), 'Analyzing the Internet as a Common Pool Resource: The Problem of Network Congestion'. pre-conference draft.

Boyle, J. (1997), 'A Politics of Intellectual Property: Environmentalism for the Net?', http://www.law.duke.edu/boylesite/intprop.htm.

Boyle, J. (2003), 'Foreword: The Opposite of Property?', *Law and Contemporary Problems* **66**.

Cahir, J. (2003), 'The Information Commons', SSRN Electronic Library, http://ssrn.com/abstract=428584. Queen Mary Intellectual Property Working Paper.

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, S. 95–154. cyberlaw.stanford.edu/blogs/cooper/archives/openarchitecture.pdf.

Denning, P. J. (2004), 'The Social Life of Innovation: The Profession of IT', *Communications of the ACM* **47**(4), S. 15–19.

18 Carol Rose beruft sich auf die Ausführungen von Hirschmann (1980, S. 67 ff.) im Kapitel „Annehmlichkeit und Unschuld des Gelderwerbs und des Handels".

Frischmann, B. M. (2004), 'An Economic Theory of Infrastructure and Sustainable Infrastructure Commons', SSRN Electronic Library, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=588424.

Hardin, G. (1968), 'The Tragedy of the Commons', *Science* **162**, S. 1243–1248.

Heller, M. A. (1998), 'The Tragedy of the Anticommons: Property in the Transition from Marx to Markets', *Harvard Law Review* **111**, S. 621–688.

Heller, M. A. und Eisenberg, R. S. (1998), 'Can Patents Deter Innovation? The Anticommons in Biomedical Research', *Science* **280**, S. 698–701.

Hess, C. und Ostrom, E. (2003), 'Ideas, Artifacts, and Facilities: Information as a Common-Pool Resource', *Law and Contemporary Problems* **66**, S. 111–146.

Hirschmann, A. O. (1980), *Leidenschaften und Interessen. Politische Begründungen des Kapitalismus vor seinem Sieg*, Suhrkamp, Frankfurt.

Kirkman, G. S., Osorio, C. A. und Sachs, J. D. (2002), The Networked Readiness Index: Measuring the Preparedness of Nations for the Networked World, *in* G. S. Kirkman, P. K. Cornelius, J. D. Sachs und K. Schwab (Hrsg.), 'The Global Information Technology Report 2001–2002: Readiness for the Networked World', Harvard University and World Economic Forum, Oxford University Press.

Lessig, L. (2001), *The Future of Ideas. The Fate of the Commons in a Connected World*, Random House, New York.

Lutterbeck, B. (1997), 'Ist das Internet ein Modell der Informationsgesellschaft?', http://ig.cs.tu-berlin.de/oldstatic/bl/024/index_html.

Mankiw, N. G. (2004), *Economics*, 3. Aufl., Thomson South-Western, Mason/Ohio.

Osterloh, M. und Rota, S. (2004), 'Open Source Software Development – just Another Case of Collective Invention?', SSRN Electronic Library, http://ssrn.com/abstract=561744.

Osterloh, M., Rota, S. und Kuster, B. (2004), Open-Source-Softwareproduktion: Ein neues Innovationsmodell?, *in* R. A. Gehring und B. Lutterbeck (Hrsg.), 'Open Source Jahrbuch 2004', Lehmanns Media, Berlin, S. 121–137.

Powell, M. K. (2004), Preserving Internet Freedom: Guiding Principles for the Industry, *in* 'At the Silicon Flatirons Symposium on „The Digital Broadband Migration: Toward a Regulatory Regime for the Internet Age"', University of Colorado School of Law, Boulder, Colorado. http://hraunfoss.fcc.gov/edocs_public/attachmatch/DOC-243556A1.doc.

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

Samuelson, P. (2001), 'Towards a New Politics of Intellectual Property', *Communications of the ACM* **44**, S. 98–99.

Schumpeter, J. (1997), *Theorie der wirtschaftlichen Entwicklung*, 9. Aufl., Dunker & Humboldt, Berlin.

Searls, D. (2002), 'Is Linux Infrastructure? Or is it Deeper than that?', *linuxjournal.com* . http://www.linuxjournal.com/article.php?sid=6074.

Solum, L. A. und Chung, M. (2003), 'The Layers Principle: Internet Architecture and the Law', SSRN Electronic Library, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=416263. University San Diego Public Law Research Paper No. 55.

Tuomi, I. (2003), *Networks of Innovation. Change and Meaning in the Age of Internet*, Oxford University Press, Oxford.

Whitt, R. S. (2004), Formulating a New Public Policy Framework Based on the Network Layers Model, *in* M. N. Cooper (Hrsg.), 'Open Architecture as Communications Policy', Center for Internet and Society, Stanford Law School, S. 353–399. cyberlaw.stanford.edu/blogs/cooper/archives/openarchitecture.pdf.

Wu, T. (2004), Broadband Policy: A User's Guide, *in* Cooper (2004), S. 233–257. in Cooper2004a and SSRN Electronic Library, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=557330.

van Schewick, B. (2004), Architecture and Innovation. The Role of the End-to-End Argument in the Original Internet, Master's thesis, Technische Universität Berlin, Berlin. Dissertation an der Fakultät für Informatik.

Kapitel 6

Open Content

„Because mutiny on the bounty's what we're all about
I'm gonna board your ship and turn it on out
No soft sucker with a parrot on his shoulder
'Cause I'm bad gettin' bolder – cold getting colder
Terrorizing suckers on the seven seas
And if you've got beef – you'll get capped in the knees
We got sixteen men on a dead man's chest
And I shot those suckers and I'll shoot the rest
(chorus) Most illingest b-boy – I got that feeling
Cause I am most ill and I'm rhymin' and stealin'"

– Beastie Boys, Rhymin' and Stealin', Licensed to Ill, 1986

Inhalte wollen frei sein

CLEMENS BRANDT

1. Einleitung zum „Kapitel“ Open Content

> „Zwei kleine Jungen tauschten Spielzeuge und beide gingen mit jeweils einem Spielzeug nach Hause. Zwei weise Männer tauschten Ideen und beide gingen mit jeweils zwei Ideen nach Hause.“ [1]

Dass Open Source nicht nur mit Computerprogrammen etwas zu tun hat, ist hinlänglich und seit langer Zeit bekannt. Open Source ist eine Herangehensweise, wie man etwas kreiert. Es ist ganz natürlich, dass Menschen Ideen austauschen, auf ihnen aufbauen und sie modifizieren. Dies wurde schon seit Urzeiten so gemacht und liegt in unserer Natur. Richard Stallman vergleicht Computerprogramme mit Kochrezepten (nach Hall 2003). Sie gehören zu denjenigen alltäglichen Dingen, die am ehesten mit Software verwandt sind. Ein Kochrezept ist eine Liste mit Befehlen und Verfahren, die ein Koch zu befolgen hat – analog zum Quellcode eines Programms. Jedoch genau wie ein Rezept, verlangt ein Computerprogramm nach Freiheit – der Freiheit des Modifizierens, des Verbesserns. Ein Kochrezept wird weitergegeben, verbessert, modifiziert und ist letztendlich Teil der Kultur einer Gesellschaft. Ähnlich zu Rezepten verhält es sich mit vielen anderen „Inhalten“. Sie sind meist nicht auf einen Schöpfer zurückzuführen, sondern ein Gemeinschaftsprozess. Selbst bei literarischen Werken erfindet ein Autor das Werk nicht von Grund auf neu – es wird stets auf bereits Niedergeschriebenem aufgebaut, stets werden Ideen ausgetauscht und weiterverarbeitet. Mit zunehmender Vernetzung der Welt erreicht dieser Austausch eine bisher nicht gekannte Blüte.

Ideen wollen frei sein[2] – dieser Auffassung ist nicht jeder. Seit dem Aufkommen des Internets wird diese Ansicht immer öfter angegriffen. Neben dem großen Berg

1 Afrikanisches Sprichwort, zitiert in Rajani (2003, S. 20).

2 Im englischen Original lautet dieser Satz „Information wants to be free“. Es ist ein häufig eingesetzter und höchst kontroverser Aphorismus, dessen Ursprung man sogar bis in die Bibel zurückverfolgen kann. Bekannt wurde er jedoch erst durch Stewart Brand, der auf der „First Hacker's Conference“ 1984 diesen Satz in einer Diskussion von sich gegeben hatte. Bekannte Verfechter dieser Auffassung, wie zum Beispiel Richard Stallman, John Perry Barlow oder James Boyle, haben den Satz in ihren Büchern und Aufsätzen verwendet (Clarke 2000).

der Erfindungen, also jenen Ideen, die dem industriellen Nutzen dienen, sind es auch die künstlerischen Inhalte, die vermehrt und verstärkt vom proprietären Modell vereinnahmt werden. „Open Content“[3] bezeichnet solche Inhalte, die gerade nicht von einer Person oder Organisation im Zaum gehalten werden, sondern bei denen die Weitergabe sowie in den meisten Fällen das Weiterverarbeiten erwünscht ist und nicht verboten sind. Dieses Kapitel befasst sich mit diesen Inhalten und stellt dieser Umgangsweise das derzeitig vorherrschende proprietäre Modell entgegen. Weiterhin werden die unterschiedlichen Ausprägungen beleuchtet und anhand erfolgreicher Projekte aus der Praxis vorgestellt, was in diesem Jahr musikalische und textliche Inhalte sind. Dieses Kapitel soll natürlich über die stetig expandierende Bewegung informieren. Es soll aber auch motivieren, selber daran teilzunehmen, da Ideen gerade von ihrem Austausch leben und davon, dass sie weiterverarbeitet werden.

2. Das „geistige Eigentum“

Nach Mark Lemley hat dieser Begriff seine eigentliche Bedeutung erst gewonnen, als die *World Intellectual Property Organization* (WIPO)[4] sich 1967 in Stockholm gegründet hat (Rajani 2003). Aufbauend auf den Berner und Pariser Verträgen unterscheidet die WIPO bei „geistigem Eigentum“ zwischen „industriellem Eigentum“ und literarischen bzw. künstlerischen Schöpfungen. Dieser Begriff bringt jedoch einige Missverständnisse mit sich, schon aus dem Grund, dass es sich um geistige Schöpfungen handelt und gerade nicht um ein Stück abgezäuntes Land, dessen Betreten verboten ist. Obwohl alleine die Geschichte über die Ursprünge des Begriffs diese Einleitung bei weitem sprengen würde, ist es notwendig, in die Vergangenheit zu schauen – vor allem auf das Land, welches heutzutage als größter Verfechter von ausgedehnten Rechten an „geistigen Eigentum“ gilt.

Thomas Jefferson, einstiger Präsident der Vereinigten Staaten von Amerika, hat 1813 in einem Brief an Isaac Mcpherson Folgendes geschrieben:

> „Wer eine Idee von mir bekommt, erhält Unterweisungen ohne die meinigen zu mindern; so wie derjenige, der seine Fackel an meiner entzündet, Licht erhält, ohne mich zu verdunkeln.“ (zitiert in Grassmuck 2003, S. 2)

3 Der Begriff „Open Content“ oder „Freie Inhalte“ ist auf David Wiley zurückzuführen. Dieser gründete 1998 die Open-Content-Initiative (siehe http://www.opencontent.org – die Seite wird jedoch gerade umgestaltet) und entwickelte auf Basis der GNU General Public License die Open Publication License (zu finden unter http://www.opencontent.org/openpub/ [15. Jan 2005]), welche die freie Weitergabe von Inhalten ermöglichen soll (Rehm und Lobin 2001, Cohen 2003).

4 Die WIPO (http://www.wipo.int/) ist eine Organisation der Vereinten Nationen mit Sitz in Genf. Sie ging aus den Berner und Pariser Verträgen hervor, die einen internationalen Standard für den Schutz von literarischen und künstlerischen Werken zum einen und von „industriellem Eigentum“ zum anderen definierten. Der von der WIPO initiierte *Patent Copyright Treaty* (PCT) von 1970 sowie der *WIPO Copyright Treaty* (WCT) von 1996 sind die Grundlage für heutige Gesetze, die den Schutz von „geistigem Eigentum“ regeln. Auf Basis des WCT wurde unter anderem in den USA der *Digital Millenium Copyrights Act* (DMCA) 1998 implementiert (Long et al. 1996).

Thomas Jefferson hat mit den Weg geebnet, der die USA zu der mächtigsten Nation dieser Welt machte. Ein Teil dieses Weges war der Umgang mit „geistigem Eigentum". Es war hauptsächlich der freie Umgang mit Ideen, der dem Land neben großem Wohlstand auch eine immense kulturelle Vielfalt bereitet hat. Was beim Urheberrecht einst mit einem 14 Jahre währenden Schutz von literarischen Werken begann,[5] umfasst heute nahezu sämtliche kreative Werke und geht erst 70 Jahre nach dem Tod des Schöpfers in die *Public Domain* über.[6] Diese Entwicklung betrifft nicht nur die USA. Mit dem *WIPO Copyright Treaty* haben sich 180 Staaten dazu verpflichtet, einen starken Schutz der „geistigen Eigentumsrechte" zu garantieren (WIPO 2004). Was einst die USA geformt hat, bleibt nun vielen Staaten verwehrt.

Der vermeintliche proprietäre Charakter von Inhalten hat sich als feste Instanz in unser westliches Wertesystem geschlichen. Siva Vaidhyanathan, Dozent an der New York University, beschreibt im ersten Artikel „Open Source as Culture—Culture as Open Source" diesen Prozess. Der Umgang mit „geistigem Eigentum" war eigentlich schon immer ein freier Umgang, der mit dem Open-Source-Modell vergleichbar ist. Das proprietäre Modell hat diesen Umgang jedoch fest vereinnahmt, wobei die Open–Source-Bewegung aber auch langsam ihre ersten Früchte außerhalb von Computerprogrammen zeigt. Elementare Kommunikationsprozesse, wie das Veröffentlichen von Texten, werden zurückerobert und verbessern durch den freien Informationsfluss den Austausch an Ideen in unserer Gesellschaft.

3. Creative Commons

Einer der prominentesten Verfechter von freien Inhalten, Lawrence Lessig, ist der Auffassung, dass

> „. . .the reason perfect control has not been our tradition's aim is that creation always involves building upon something else. There is no art that doesn't reuse. And there will be less art if every reuse is taxed by the appropriator. Monopoly controls have been the exception in free societies; they have been the rule in closed societies." (Zitiert in Lessig 2001)

Um den immer enger werdenden Schnüren des Urheberrechts entgegenzutreten, hat Lawrence Lessig zusammen mit anderen Rechtsexperten im Jahre 2000 die Creative-Commons-Initiative (CC) ins Leben gerufen. Ziel dieser Initiative ist es,

5 1790 hat der Kongress der USA den ersten *Copyright Act* verabschiedet. Dieser umfasste nur literarische Publikationen und war von 14 Jahren Dauer. Bei Bedarf konnte diese Frist bei Ablauf um den selben Zeitraum verlängert werden. Literarische Werke mussten aber erst angemeldet werden, bevor sie unter Schutz standen. In den ersten zehn Jahren wurden fünf Prozent aller Werke unter *Copyright* gestellt, ausländische Publikationen waren nicht geschützt (Lessig 2004).

6 1998 verabschiedete der Kongress der USA den *Sonny Bono Copyright Extension Act* (benannt nach dessen Initiator), der das Recht an einem Werk eines Autors auf dessen Lebenszeit, sowie weiteren 70 Jahren ansetzt. Zudem erhalten bereits veröffentlichte Werke einen weiteren Aufschub von 20 Jahren. Auftragswerke werden für 95 Jahre nach dem Erscheinen oder, falls es kürzer sein sollte, für 120 Jahre nach dem Kreieren geschützt (U.S. Congress 1998).

Künstlern, Autoren und generell Schöpfern von kreativen Inhalten eine rechtliche Plattform zu bieten, sodass sie selber entscheiden können, welche Rechte auf ihren Werken liegen. Inspiriert von der GNU General Public License (GPL) wurden die Creative-Commons-Lizenzen geschaffen, welche sich nicht auf Software spezialisieren, sondern gerade auf andere Inhalte wie Literatur, Musik, Film, Webseiten usw. (Creative Commons 2004).

Die CC-Lizenzen bauen, wie die GPL, auf dem Urheberrecht auf. Im Gegensatz zur GPL, bei der ad hoc auf jedem Werk „alle Rechte vorbehalten" sind, verhält es sich bei CC-Lizenzen anders – es sind „manche Rechte vorbehalten". Der Kern besteht aus den drei Gebieten: Namensnennung, kommerzieller Nutzen und abgeleitete Werke. Man kann als Schöpfer also auswählen, ob andere das Werk tauschen (solange es einem zugute gehalten wird), es weiterverarbeiten oder kommerziellen Nutzen daraus ziehen dürfen. Ein weiteres Ziel der Lizenzen ist, dass sie möglichst einfach eingesetzt werden können. Auf der Website[7] kann man sich seine gewünschte Lizenz erstellen und erhält automatisch einen einfach verständlichen Text, ein juristisches Dokument und einen *RDF Code* der Lizenz generiert. Aufgrund dieser sehr guten Indizierbarkeit von Werken, ist die Lizenz besonders für Inhalte im Internet geeignet, obwohl sie auch vermehrt für konventionelle Medien verwendet wird wie zum Beispiel für die meisten Artikel in diesem Buch (Creative Commons 2004).

Zu den Standard-Lizenzen wurden noch weitere geschaffen, wie z. B. 2004 die Sampling-Lizenz[8]. *Sampling* bedeutet, dass man wenige Takte, manchmal sogar nur einzelne Klänge von Instrumenten, aus bestehenden Liedern herausschneidet und sie in neuen Stücken kombiniert. Seit dem Aufkommen von Technologien, die solch ein Ausschneiden von Musik ermöglichen, sind neue Musikrichtungen wie „HipHop" und „Techno" entstanden. Alle würden in ihrer jetzigen Form ohne *Sampling* nicht bestehen. Jedoch ist heutzutage nicht mehr die Technologie das teure Element, sondern vielmehr das Abklären von Rechten. Selbst wenn Musiker gerne bereit sind, ihre Musik zum Sampeln freizugeben, haben die Plattenfirmen meist andere Interessen. Dieses Problem lösen die CC-Sampling-Lizenzen, bei denen der Musiker selbst entscheiden kann, in welchem Grad seine Musik weiterverwendet werden soll. Im November 2004 erschien im Magazin *Wired* bereits die erste CD mit Liedern unter dieser Lizenz.[9]

Neben der Spezialisierung der Lizenzen breiten sich diese auch immer weiter auf der Welt aus. Es existieren bereits Lizenzen für 13 Staaten, allesamt mit Anpassungen auf das jeweilige Rechtssystem. 2004 wurde beim *Wizards of OS*-(WOS) Kongress[10] die deutsche Version der Creative Commons eingeführt. Neben den unzähligen Webseiten, die CC-Lizenzen nutzen, gibt es auch äußerst ambitionierte Projekte, wie das *Creative Archive* der BBC, welches sämtliche Fernsehsendungen nach ihrer Ausstrahlung unter CC-Lizenz in einem Archiv speichert (Krüger 2004). Außerdem nutzen Netlabels die Lizenzen, um Musik im Internet zu veröffentlichen. Autoren wie Cory

7 http://creativecommons.org
8 http://creativecommons.org/license/sampling?format=audio
9 http://wired.com/wired/archive/12.11/sample
10 http://www.wizards-of-os.org/

Doctorow[11] und nicht zuletzt Lawrence Lessig selber[12] veröffentlichen Bücher unter CC-Lizenz.

4. Gefangene Musik – Befreite Musik

Musikinteressierten ist das Thema „mp3 und Musikindustrie" mittlerweile schon mehr als leid. Wie oft wurde bereits ein jäher Untergang jener Industrie prophezeit und es gibt sie immer noch – sogar stärker etabliert als in den letzten Jahren zuvor. Besonders in den USA und England konnte die Musikindustrie eine Steigerung des Umsatzes verbuchen.[13]

Auch der Verkauf von Musik über Online-Portale wird immer erfolgreicher. Laut dem „Digital Music Report 2005" der *International Federation of the Phonographic Industry* (2005) betrugen die Gesamtumsätze des letzten Jahres 330 Millionen US–Dollar, wobei für dieses Jahr noch eine Steigerung um das Doppelte erwartet wird, welche nicht nur durch die „lieb gewonnenen" Handy-Klingeltöne erreicht wird. Hierzulande hielt sich die Begeisterung für die Portale der großen Plattenfirmen eher in Grenzen. Nach großem Aufmarsch und einem begeisterten Bundeskanzler ging das Online-Vertriebsportal der deutschen Musikindustrie *Phonoline*[14] sang- und klanglos wieder unter (Kuri 2004). Ähnlich erging es dem daran angeschlossenen Portal von Universal Music *Popfile*. Anders verhält es sich jedoch mit der Firma Apple, welche mit der Kombination *iPod* und dem *iTunes Music Store* hohe Gewinne erzielt und auch hier zu Lande an erster Stelle mit der Zahl der Downloads steht (N24.de 2004).

In dem „Digital Music Report 2005" werden, wie nicht anders zu erwarten, auch Tauschbörsen erwähnt und als Quelle allen Unheils dämonisiert. Die Musikindustrie sieht im „illegalen Downloaden" ihren größten Feind – nicht selten werden Wörter wie „Internetpiraterie" oder „Diebstahl" verwendet. Die großen Plattenfirmen wenden sämtliche Mittel an, um diesem „Problem" Herr zu werden. Da seit März 2004 mit der *Richtlinie über die Maßnahmen und Verfahren zum Schutz der Rechte an geistigem Eigentum* und deren Implementierung in den Gesetzen der Mitgliedstaaten auch in Europa eine geeignete Grundlage besteht,[15] fängt die Musikindustrie bereits eifrig an, Tauschbörsen-Nutzer zu verklagen (Wilkens 2004).

Die Verschärfungen sind hauptsächlich auf die gute Lobbyarbeit der Verbände der Content-Industrie zurückzuführen. Mit dem *WIPO Copyright Treaty* sowie dem

11 Die Bücher des Science-Fiction Autoren Cory Doctorow kann man von seiner Website http://www.craphound.com/ herunterladen.

12 Das Buch „Free Culture" ist unter CC-Lizenz auf http://www.free-culture.cc/ veröffentlicht.

13 Siehe hierzu die Berichte der *Recording Industrie Association of America* (RIAA, http://www.riaa.com/news/newsletter/102004.asp) und der *British record industry's trade association* (BPI, http://www.bpi.co.uk/pdf/media_q304.pdf).

14 Phonoline ist ein Gemeinschaftsprojekt der deutschen Musikindustrie und von T-Com. Es ist ein Portal für den Online-Vertrieb von Musik und bietet somit eine technische Grundlage für Internet-Musikshops, wie dem *Coca-Cola Music Store* oder *Popfile* von Universal.

15 Die sich an die Richtlinie haltenden Gesetze bieten Haltern von Urheberrechten die Möglichkeit, auf einfachem Weg gegen Tauschbörsennutzer vorzugehen. Das bedeutet, dass ohne komplizierte Strafverfahren umfassende Auskünfte über Personen eingeholt werden können, die des Verstoßes gegen das Urheberrecht verdächtigt werden (Krempl 2004*a*).

TRIPS-Übereinkommen[16] der *World Trade Organization* (WTO) setzt die Industrie ihre Arbeit auf höchster, internationaler Ebene an, was sich wiederum auf nationales Recht weitervererbt. Wie die *Gartner Group*[17] und das *Berkman Center for Internet and Society*[18] in ihrem Bericht „Copyright and Digital Media in a Post-Napster World" (2005*a*) feststellen, wird die Kluft der Interessen zwischen den Rechteverwaltern und den Verbrauchern in den kommenden Jahren sogar noch größer werden. Derzeit stehen in den USA mehrere Gesetze zur Debatte, die entweder für mehr Verbraucherrechte oder eine Verschärfung des Schutzes von „geistigem Eigentum" eintreten. Es werden vor allem aber die zukünftigen Verträge der WTO und WIPO sein, die eine Richtung für den Umgang mit Urheberrechten festlegen werden (Gartner and Berkman 2005b).

Zusätzlich zu den rechtlichen Mitteln probiert die Content-Industrie auch über technische Mittel, ihr Geschäftsmodell am Leben zu erhalten. Ein häufig diskutiertes Thema ist derzeit *Digital Rights Management* (DRM). Es bezeichnet einen Mechanismus, der das Verwenden von digitalen Werken reguliert. Vertreiber von Inhalten können über technische Schranken bestimmen, wie oft man die Datei zum Beispiel kopieren, benutzen oder per E-Mail versenden kann. Dass dies einige Probleme mit sich bringt, wissen auch Tile von Damm, Jens Herrmann und Jan Schallaböck, die in ihrem Artikel „Industrial Influences" einen kritischen Blick auf die derzeitige Situation des Musikmarktes werfen. Neben der Beleuchtung verschiedener Modelle für den Umgang mit musikalischen Inhalten, betrachten sie auch die erwähnte DRM-Problematik. Dabei stellen sie fest, dass die derzeitige Entwicklung fatale Folgen für Open Source und den freien Informationsfluss im Allgemeinen haben kann.

Aber genau dieser freie Informationsfluss eckt mit dem proprietären Modell der Content-Industrie an. Dabei sieht sich nicht nur die Musikindustrie durch den freien Austausch von Inhalten bedroht; auch die Filmindustrie hat im letzten Jahr mit ihrer „Hart aber gerecht"-Werbekampagne[19] einiges an Aufmerksamkeit auf sich gezogen. Wie sie selber behauptet, möchte sie nicht so „lax" mit dem „Problem" umgehen wie die Musikindustrie (Krempl 2003). Die Drohungen der Content-Industrie scheinen die Bevölkerung jedoch nicht davon abzuschrecken, weiterhin Inhalte über das Internet frei zu tauschen. In Deutschland hat sich im letzten Jahr die Zahl der „illegalen" Musikdownloads verdoppelt (Allensbach 2004) und auch im Jahr 2005 wird dieser Trend nicht aufzuhalten sein.

Es bleibt also einiges für 2005 zu erwarten: mehr Klagen, mehr Downloads, sich verändernde technische Infrastrukturen und eine höhere Konzentration des Marktes der Content-Anbieter.[20]. *„Niemals in unserer Geschichte befanden sich mehr Kontrollrechte in den Händen so weniger Menschen"*, so Lawrence Lessig auf dem WOS Kongress 2004 in

16 *TRIPS* steht für *Trade-Related Aspects of Intellectual Property Rights* (http://www.wto.org/english/tratop_e/trips_e/trips_e.htm).

17 http://www.g2.com/

18 http://cyber.law.harvard.edu/

19 http://www.hartabergerecht.de/

20 Zum Beispiel ist seit 2004 die Fusion von Sony Music und BMG wirksam, durch welche die Konzerne nach Universal Music zum zweitgrößten Musikkonzern gewachsen sind (Löding 2004).

Berlin (zitiert nach Krempl 2004*b*). Es bleibt also nichts anderes übrig, als das Schwert selbst in die Hand zu nehmen – gerade durch die Creative-Commons-Lizenzen ist eine fruchtbare Grundlage für freie Inhalte gegeben.

Ein solcher Kämpfer ist Sebastian Redenz mit seinem Zugpferd, dem *Thinner Netlabel*[21]. Unter CC-Lizenzen veröffentlicht er elektronische Musik – frei von digitalen Beschränkungen und zum kostenlosen Download. In seinem Artikel „Das Netlabel als alternativer Ansatz der Musikdistribution" erklärt er, wie ein Netlabel funktioniert und inwieweit es sich von einem „koventionellem" Label unterscheidet. Des Weiteren bietet er uns einen Einblick und Ausblick in eine alternative Form der Musikdistribution, fern von CDs und DRM.

5. Kooperative Inhalte

Neben Filmen und Musik sind es aber auch Texte, die vermehrt vom Urheberrecht gebrandmarkt werden. Für eine Demokratie ist es von äußerster Wichtigkeit, dass Informationen frei zugänglich sind. Ohne diesen freien Zugang ist freie Meinungsäußerung kaum möglich. Ebenso bedeutend ist jedoch auch, dass diese Informationen weiterverarbeitet werden dürfen, gerade damit Meinungen und Ansichten darüber überhaupt offen kund getan werden können. Wenn proprietäre Rechte auf Inhalte jedoch zu ausgedehnt werden, sodass keine Weiterverarbeitung, insbesondere für nicht-kommerzielle Zwecke, mehr möglich ist, dann ist ein stützender Pfeiler unserer Demokratie in Gefahr.

In den letzten Jahren hat sich im Internet ein alternatives Modell zum Veröffentlichen von Informationen entwickelt. So genannte „Weblogs", kurz „Blogs", sind Informationssysteme, die Hypertext mit Diskussionsforen verbinden, in denen periodisch Einträge unterschiedlichster Themen veröffentlicht werden. Dabei handelt es sich nicht nur um ein selbstverliebtes Portrait oder digitales Tagebuch – es sind kollaborative Publikationen, die durchaus eine Konkurrenz zu Onlineangeboten mancher Publikationen sein können.[22] Blogs zeichnen sich dadurch aus, dass sie hochgradig mit anderen Inhalten verlinken und auf ihnen aufbauen. Dabei können sie von einem Nutzer, von einer Gruppe oder sogar von jedem, der die Seite betritt, geführt werden (Kahn und Kellner 2003).

Besonders für politisch interessierte und aktive Menschen haben sich Blogs als sehr gutes Medium zum Informationsaustausch herausgestellt. Entgegen dem Vorurteil, dass sie hauptsächlich von Personen des linken Spektrums genutzt werden, *bloggen* Menschen jeglicher Couleur, verlinken sich untereinander und nehmen Bezug auf Geschehnisse des Tages. So zum Beispiel auch die Seite http://www.freerepublic.com, auf der Anhänger des rechten Spektrums der politischen Landschaft in den USA Nachrichten publizieren, die meist auf Zeitungsartikeln aufbauen und diese häufig kritisieren. Es ist schon bedeutend, dass gerade die von der Seite häufig kritisierte „liberale Presse" eine Klage wegen Urheberrechtsverletzung eingereicht hat. So er-

21 http://www.thinnerism.com/

22 Als bekanntes Beispiel ist http://www.boingboing.net zu nennen.

geht es nicht nur *freerepublic.com* – das Regime des „geistigen Eigentums" bedroht fundamentale demokratische Prozesse (Pfaffenberger 2001).

Doch auch hier helfen Open-Content-Lizenzen wie Creative Commons oder die GNU Free Documentation License. Aufbauend auf diesen Lizenzen können freie Inhalte geschaffen werden, die von jedem weiterverarbeitet werden dürfen und frei ausgetauscht werden können. Eines der bisher erfolgreichsten Open-Content-Projekte ist die Internet-Enzyklopädie *Wikipedia*[23]. Sie basiert auf dem „Wiki-Prinzip", was bedeutet, dass jeder ohne Anmeldung oder Filter Inhalte auf einer Seite veröffentlichen kann. Demnach kann bei Wikipedia jede Person Enzyklopädie-Artikel erstellen oder auch bearbeiten – im Gegensatz zu Seiten wie *Slashdot*[24], auf denen Redakteure die eingetragenen Beiträge filtern und dann erst der Öffentlichkeit zugänglich machen. Wikipedia umfasst mittlerweile mehr als eine Million Artikel und ist somit größer als jede andere Enzyklopädie. Jakob Voss und Patrick Dannowski, beide sind aktive „Wikipedianer", stellen das Projekt in ihrem Artikel „Das Wissen der Welt – Die Wikipedia" umfassend vor und erläutern die Infrastruktur, welche diese großartige Sammlung erst ermöglicht hat.

6. Zusammenfassung

Die in den Artikeln besprochenen Beispiele lassen auf mehr hoffen. Jeden Tag entstehen neue Inhalte in kollaborativen Prozessen. Blogs revolutionieren das Zeitungswesen, Netlabels mischen die verkrustete Musiklandschaft auf, Projekte wie Wikipedia zeigen, wie erfolgreich Gemeinschaftswerke sein können und wieviel wiederum die Gemeinschaft von ihnen profitieren kann.

Der beschriebene Prozess ist ein *bottom-up*-Prozess. Gerade dies macht ihn so erfolgreich, aber auch in höchstem Maße von dem Grad der Partizipation abhängig. Dieses Kapitel soll motivieren – zum einen, sich mit diesem Prozess zu beschäftigen, zum anderen aber auch, an diesem Prozess aktiv teilzunehmen. Es ist Zeit, die Welt der unbegrenzten Inhalte näher kennen zu lernen.

Literaturverzeichnis

Clarke, R. (2000), 'Information Wants to be Free . . .',
http://www.anu.edu.au/people/Roger.Clarke/II/IWtbF.html [18. Jan 2005].

Cohen, N. (2003), 'OPEN CONTENT: THE REVOLUTION IN PUBLISHING – Will Open Source evangelist Bruce Perens upset the way books are sold?',
http://www.open-mag.com/features/Vol_49/Perens/Perens.htm [18. Jan 2005].

Creative Commons (2004), 'Frequently Asked Questions', http://creativecommons.org/faq [18. Jan 2005].

23 http://www.wikipedia.de
24 http://slashdot.org/

Gartner|G2 and the Berkman Center for Internet and Society at Harvard Lawschool (2005*a*), 'Copyright and Digital Media in a Post-Napster World', http://cyber.law.harvard.edu/media/files/wp2005.pdf [18. Jan 2005]. updated white paper.

Gartner|G2 and the Berkman Center for Internet and Society at Harvard Lawschool (2005*b*), 'Copyright and Digital Media in a Post-Napster World: International Supplement', http://cyber.law.harvard.edu/media/files/wpsupplement2005.pdf [18. Jan 2005]. updated white paper.

Grassmuck, V. (2003), Reputation in Freier Software und Wissenschaft, *in* 'Kathedrale und Bazar: Das Phänomen „Freie Software"', Theodor-Heuss-Akademie, Gummersbach. http://fsf.pestilenz.org/volker/reputation.pdf [18. Jan 2005].

Hall, M. (2003), 'From Free Software to Open Content: An Overview of Copyleft', White paper http://www.puddingbowl.org/~mph/law_copyleft.pdf [18. Jan 2005].

Institut für Demoskopie Allensbach (2004), 'MUSIK AUS DEM INTERNET – Downloads aus dem Netz haben sich innerhalb eines Jahres mehr als verdoppelt', Allensbacher Berichte http://www.ifd-allensbach.de/pdf/prd_0419.pdf [18. Jan 2005]. Nr. 19.

International Federation of the Phonographic Industry (IFPI) (2005), 'Digital Music Report', http://www.ifpi.org/site-content/library/digital-music-report-2005.pdf [18. Jan 2005].

Kahn, R. und Kellner, D. (2003), New Media, Internet Activism, and Blogging, *in* 'The Relationship between Theory and Method in Educational Research', The University of Ljubljana. http://www.ff.uni-lj.si/pedagogika/relationship/NEW%20MEDIA,%20INTERNET%20ACTIVISM,%20AND%20BLOGGING.pdf [18. Jan 2005].

Krempl, S. (2003), 'Filmwirtschaft startet Abschreckungskampagne gegen Raubkopierer', heise online http://www.heise.de/newsticker/meldung/42431 [18. Jan 2005].

Krempl, S. (2004*a*), 'Copyright-Krieg in der EU', *Telepolis* . http://www.telepolis.de/deutsch/special/med/6344/1.html [18. Jan 2005].

Krempl, S. (2004*b*), 'WOS3: Creative Commons als Geheimwaffe der Künstler im Copyright-Krieg', heise online http://www.heise.de/newsticker/meldung/48184 [18. Jan 2005].

Krüger, A. (2004), 'Internet-TV auf Abruf', *Telepolis* . http://www.heise.de/tp/r4/artikel/17/17363/1.html [18. Jan 2005].

Kuri, J. (2004), 'Musikindustrie will Online-Plattform Phonoline aufgeben [Update]', heise online http://www.heise.de/newsticker/meldung/51482 [18. Jan 2005].

Lessig, L. (2001), 'May the source be with you', *Wired* **9**(12). Wired http://www.wired.com/wired/archive/9.12/lessig.html [18. Jan 2005].

Lessig, L. (2004), *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, The Penguin Press, New York, NY. http://www.free-culture.cc/freeculture.pdf [18. Jan 2005].

Long, D. E. et al. (1996), Treaty Regimes, *in* A. D'Amato und D. E. Long (Hrsg.), 'International Intellectual Property Anthology', Anderson Publishing, Cincinnati, Ohio, Kapitel 7, S. 189–242.

Löding, T. (2004), 'Bertelsmann und Sony: Musikfusion ist perfekt', heise online http://www.heise.de/newsticker/meldung/49848 [18. Jan 2005].

N24.de (2004), 'Andrang an der Musiktankstelle – Kommerzielle Downloads versprühen Sex-Appeal', N24.de http://www.n24.de/wirtschaft/branchen/index.php/a2004111011160749952 [18. Jan 2005].

Pfaffenberger, B. (2001), 'Why Open Content Matters', *Linux Journal* . Linux Journal http://www.linuxjournal.com/article/4709 [18. Jan 2005].

Rajani, N. (2003), 'Free as in Education – Significance of the Free/Libre and Open Source Softwarefor Developing Countries', White paper for the Ministry of Foreign Affairs – Finland http: //www.itu.int/wsis/docs/background/themes/access/free_as_in_education_niranjan.pdf [18. Jan 2005].

Rehm, G. und Lobin, H. (2001), 'From Open Source to Open Information: Collaborative Methods in Creating XML-based Markup Languages', http://www.uni-giessen.de/~g91063/pdf/open-information.pdf [18. Jan 2005].

U.S. Congress (1998), 'Sonny Bono Copyright Term Extension Act'. S. 505, http://lcweb.loc.gov/copyright/legislation/s505.pdf [18. Jan 2005].

Wilkens, A. (2004), 'Europäische Musikindustrie startet Klagewelle', heise online http://www.heise.de/newsticker/meldung/51909 [18. Jan 2005].

World Intellectual Property Organization (WIPO) (2004), 'General Information'. http://www.wipo.int/freepublications/en/general/400/wipo_pub_400.pdf [18. Jan 2005].

Open Source as Culture—Culture as Open Source

Siva Vaidhyanathan

The Open Source model of peer production, sharing, revision, and peer review has distilled and labeled the most successful human creative habits into a techno- political movement. This distillation has had costs and benefits. It has been difficult to court mainstream acceptance for such a tangle of seemingly technical ideas when its chief advocates have been hackers and academics. On the other hand, the brilliant success of overtly labeled Open Source experiments, coupled with the horror stories of attempts to protect the proprietary model of cultural production have served to popularize the ideas championed by the movement. In recent years, we have seen the Open Source model overtly mimicked within domains of culture quite distinct from computer software. Rather than being revolutionary, this movement is quite conservatively recapturing and revalorizing the basic human communicative and cultural processes that have generated many good things.

The "Open Source" way of doing things is all the rage. Companies as powerful and established as IBM boast of using Linux operating systems in servers. Publications as conservative as *The Economist* have pronounced Open Source methods *successful* and have pondered their applicability to areas of research and development as different from software as pharmaceutical research (see Economist 2004, Weber 2004).

It is striking that we have to employ phrases like "Open Source" and "Free Software" at all.[1] They are significant, powerful phrases simply because they represent an insurgent model of commercial activity and information policy. They challenge the entrenched status quo: the proprietary model of cultural and technological production.

But this has only recently been the case. The "Open Source" way is closer to how human creativity has always worked. Open Source used to be the default way

1 Throughout this essay and in all of my work I intentionally conflate these two terms while being fully aware of the political distinction that Richard Stallman emphasizes in his defense of "Free Software". Stallman's point – that "Open Source" invites an emphasis on convenience and utility rather than freedom and community, was important to make in the 1990s. He lost the battle to control the terms, just as he has had to concede the rhetorical convenience and ubiquity of "Linux" instead of the more accurate "GNU/Linux". I am confident that anyone who peers into the history or politics of the Open Source movement will encounter Stallman's persuasive case for freedom and the GNU project's central contribution to the growth of the operating system we now call Linux (see Stallman 1999).

of doing things. The rapid adoption of proprietary information has been so intense and influential since the 1980s that we hardly remember another way or another time. However, throughout most of human history all information technologies and almost all technologies have been "open source". And we have done pretty well as a species with tools and habits unencumbered by high restrictions on sharing, copying, customizing, and improving.

We have become so inured by the proprietary model, so dazzled and intimidated by its cultural and political power, that any common sense challenge to its assumptions and tenets seems radical, idealistic, or dangerous. But in recent years the practical advantages of the "Open Source" model of creativity and commerce have become clear. The resulting clamor about the advantages and threats of Open Source models have revealed serious faults in the chief regulatory system that governs global flows of culture and information: copyright.

The Rise of Proprietarianism

Copyright gets stretched way out of shape to accommodate proprietary software. Copyright was originally designed to protect books, charts, and maps. Later, court rulings and legislatures expanded to include recorded music, film, video, translations, public performance, and finally practically all media that now exist or have yet to be created. Software is special, though. It's not just expression. It's functional. It's not just information. It's action. In some ways, the inclusion of software among the copyrightable forms of creativity has complicated and challenged the intellectual property tradition. Copyright and proprietary software have metastasized synergistically.

The proprietary model of software production arose sometime in the 1970s, when mainframe software vendors like AT&T and Digital started asserting control over their source code, thus limiting what computer scientists could do to customize their tools. This was an insult to and offense against these scientists who were acclimated to the academic and scientific ideologies that privilege openness and non-monetary reward systems. In a much more precise sense we can date the spark of the conflagration between the then-insurgent proprietary model and the then-dominant hacker culture (Open Source, although this term did not yet exist) to Bill Gates' 1976 open letter to the small but growing community of personal computer hackers. Gates warned them that his new company, then spelled *Micro-Soft*, would aggressively assert its intellectual property claims against those who would trade tapes carrying the company's software. Since that date, despite frequently exploiting the gaps and safety valves of copyright protection on its rise to the heights of wealth and power, Microsoft and Gates have worked in correlation if not coordination with the steady valorization of intellectual property rights as the chief locus of worldwide cultural and industrial policy (see Vaidhyanathan 2001, Wayner 2000, Raymond 1999).

According to the proprietary ideology, innovation would not occur without a strong incentive system for the innovator to exploit for commercial gain. *Fencing off* innovations becomes essential for firms and actors to establish markets and bargain

away rights. Because innovation so often concerns the ephemeral, trade regarding innovation concerns excluding other from using, exploiting, or copying data, designs, or algorithms. The Clinton, Bush, and Blair administrations in the United States and the United Kingdom embraced the proprietary model as the key to thriving through the de-industrialization of the developed world, thus locking in the advantages that educated, wired nation-states have over those that have been held in technological and economic bondage for centuries. Proprietary models of innovation policy and market relations can be powerful: witness the remarkable success and wealth of the global pharmaceutical industry, or, for that matter, Microsoft. But these models can be just as powerful with limitations that allow for communal creation, revision, criticism, and adaptability: witness the culture of custom cars or the World Wide Web (see Vaidhyanathan 2004, Lessig 2001, 2004).

In fact, as economist Richard Adkisson argues, the veneration of forceful intellectual property rights as the foundation of innovation and creativity above all other forms has promoted an unhealthy cultural and social condition, once which can generate suboptimal levels of investment, asset allocation, and policy choices. Adkisson indicts the widespread belief that intellectual property rights are the best (perhaps only) of all possible arrangements for innovation by alerting us to the *ceremonial status* these rights have assumed. *"Ceremonial encapsulation occurs when ceremonial values are allowed to alter or otherwise limit the application of technologies instrumental in the process of social problem solving,"* Adkisson writes. Specifically, Adkisson warns that blind faith in high levels of intellectual property protection is of the *future-binding type*, in which technology and mythology act synergistically to legitimize elite control over technologies or other innovative or creative processes (Adkisson 2004).

The Return of the Jedi

Richard Stallman took a stand against the proprietary model long before the rest of us even realized its power and trajectory. A computer scientist working in the 1970s and 1980s for the Artificial Intelligence project at *MIT*, Stallman grew frustrated that computer companies were denying him and other hackers access to their source code. Stallman found he was not allowed to improve the software and devices that he had to work with, even when they did not function very well. More importantly, Stallman grew alarmed that he was becoming contractually bound to be selfish and unkind. The user agreements that accompanied proprietary software forbade him from sharing his tools and techniques with others. As a scientist, he was offended that openness was being criminalized. As a citizen, he was concerned that freedoms of speech and creativity were being constricted. As a problem solver, he set out to establish the *Free Software Foundation* to prove that good tools and technologies could emerge from a community of concerned creators. Leveraging the communicative power of technology newsletters and the postal system, Stallman sold tapes with his free (as in liberated) software on them. By the time enough of his constituency had connected themselves through the Internet, he started coordinating projects and conversations among a diverse and distributed set of programmers (Stallman 1999, Williams 2002).

During the late 1990s a growing team of hackers struggled to build the holy grail of free software: an operating system kernel that would allow an array of programs to work in coordination. The group, let by Linus Torvalds, created a system that became known as Linux. It has since become the chief threat to the ubiquity and dominance of Microsoft (see Torvalds 2003, Raymond 2001).

While Linux and the GNU (Free Software) project have garnered the most attention in accounts of Open Source development, the protocols and programs that enable and empower the e-mail, the World Wide Web, IRC, and just about every other activity on the Internet all emerged from community-based project teams, often ad-hoc and amateur. The resulting protocols are elegant, efficient, effective, and under constant revision. They have empowered both the growth of the proprietary model and the Open Source model of cultural production to reach expansive new markets and audiences (see Bradner 1999, Galloway 2004).

Each of these projects illuminates what Yochai Benkler calls *peer production*. Benkler writes:

> "The emergence of free software as a substantial force in the software development world poses a puzzle for this organization theory. Free software projects do not rely either on markets or on managerial hierarchies to organize production. Programmers do not generally participate in a project because someone who is their boss instructed them, though some do. They do not generally participate in a project because someone offers them a price, though some participants do focus on long-term appropriation through money-oriented activities, like consulting or service contracts. But the critical mass of participation in projects cannot be explained by the direct presence of a command, a price, or even a future monetary return, particularly in the all- important microlevel decisions regarding selection of projects to which participants contribute. In other words, programmers participate in free software projects without following the normal signals generated by market- based, firm-based, or hybrid models." (Benkler 2002)

Economists assumed for decades that firms emerged to lower or eliminate transaction costs and coordination problems. But as it turns out, fast, efficient and dependable communication, guided by protocols both social and digital (a process Benkler calls *integration*), can generate brilliant and powerful tools and expressions. Benkler concludes:

> "The strength of peer production is in matching human capital to information inputs to produce new information goods. Strong intellectual property rights inefficiently shrink the universe of existing information inputs that can be subjected to this process. Instead, owned inputs will be limited to human capital with which the owner of the input has a contractual—usually employment—relationship. Moreover, the entire universe of peer-produced information gains no benefit from

> strong intellectual property rights. Since the core of commons-based peer production entails provisioning without direct appropriation and since indirect appropriation—intrinsic or extrinsic—does not rely on control of the information but on its widest possible availability, intellectual property offers no gain, only loss, to peer production. While it is true that free software currently uses copyright-based licensing to prevent certain kinds of defection from peer production processes, that strategy is needed only as a form of institutional jujitsu to defend from intellectual property. A complete absence of property in the software domain would be at least as congenial to free software development as the condition where property exists, but copyright permits free software projects to use licensing to defend themselves from defection. The same protection from defection might be provided by other means as well, such as creating simple public mechanisms for contributing one's work in a way that makes it unsusceptible to downstream appropriation—a conservancy of sorts. Regulators concerned with fostering innovation may better direct their efforts toward providing the institutional tools that would help thousands of people to collaborate without appropriating their joint product, making the information they produce freely available rather than spending their efforts to increase the scope and sophistication of the mechanisms for private appropriation of this public good as they now do." (Benkler 2002)

Benkler's prescriptions seem like predictions. In recent years the governments of nation-states as diverse as South Africa, Brazil, and the Peoples' Republic of China have adopted policies that would encourage the dissemination of Open Source Software.

More significantly, the Open Source model has moved far beyond software. Musician and composer Gilberto Gil, the culture minister of Brazil, has released several albums under a *Creative Commons* license. Such licenses (under which this paper lies as well) are based on the *GNU General Public License*, which "locks" the content open. It requires all users of the copyrighted material to conform to terms that encourage sharing and building (Dibell 2004).

Other significant extra-software projects based on the Open Source model include *Wikipedia*, a remarkable compilation of fact and analysis written and reviewed by a committed team of peers placed around the world. The scientific spheres have rediscovered their commitment to openness through the movement to establish and maintain Open Access Journals, thus evading the proprietary traps (and expenses) of large commercial journal publishers (Kaiser 2004). By 2004, citizen-based journalism, often known as *Open Source Journalism* grew in importance and established itself as essential element of the global information ecosystem (see Rosen 2004, Gillmor 2004). Such experiments are sure to proliferate in response to the failures (market and otherwise) of proprietary media forms (Kelty 2004).

How Open Source Changes Copyright

Copyright is a limited monopoly, granted by the state, meant to foster creativity by generating a system of presumed incentives. The copyright holder must have enough faith in the system to justify her investment. The copyright holder's rights to exclude are limited by some public values such as education and criticism. This is the standard understanding of copyright law's role and scope. But while acknowledging the interests of the public, it omits the voice of the public itself. In other words, the system cannot thrive if the public considers it to be captured, corrupted, irrelevant, or absurd (Vaidhyanathan 2004).

The rise and success of Open Source models foster a general understanding that copyright is not a single right bestowed upon one brilliant individual author, but is instead a "bundle" of rights that a copyright holder (individual, corporation, organization, or foundation) may license. Most importantly, these experiments and project show that "all rights reserved" need not be the default state of copyright protection. For many, "some rights reserved" serves the interests of creators better than the absolutist proprietary model.

As the rhetoric of Open Source and the politics of traditional knowledge and culture emerge in starker relief within the topography of copyright and cultural policy debates, their themes tend to converge. As anthropologist Vladimir Hafstein describes the tension between copyright systems as dictated by the industrialized world and modes of communal cultural production that are best (albeit not exclusively) demonstrated in developing nations, he uses terms that could just as easily be applied to technological peer production. *"Creativity as a social process is the common denominator of these concepts and approaches,"* Hafstein writes. *"From each of these perspectives, the act of creation is a social act. From the point of view of intertextuality, for example, works of literature are just as much a product of society or of discourse as they are of an individual author or, for that matter, reader."* Traditional cultural knowledge, communally composed and lacking distinct marks of individual authorship, is *"a node in a network of relations: not an isolated original, but a reproduction, a copy,"* Hafstein explains. Nothing about Hafstein's descriptions of the politics of traditional knowledge offers a resolution to that particular source of friction in global intellectual property battles. The converging rhetorics, however, reveal the extent to which innovation and creativity often (perhaps most often) lie outside the assumptions of incentives and protectionism upon which high levels of corporate copyright protection rest (see Hafstein 2004, Himanen 2001).

The Open Source model of peer production, sharing, revision, and peer review has distilled and labeled the most successful human creative habits into a political movement. This distillation has had costs and benefits. It has been difficult to court mainstream acceptance for such a tangle of seemingly technical ideas when its chief advocates have been hackers and academics. Neither class has much power or influence in the modern global economy or among centers of policy decision-making. On the other hand, the brilliant success of overtly labeled Open Source experiments, coupled with the horror stories of attempts to protect the proprietary model have added common sense to the toolbox of these advocates.

Bibliography

Adkisson, R. (2004), 'Ceremonialism, Intellectual Property Rights, and Innovation Activity', *Journal of Economic Issues* **28**(2).

Benkler, Y. (2002), 'Coase's Penguin, or, Linux and the Nature of the Firm', *Yale Law Journal* **112**(3).

Bradner, S. (1999), The Internet Engineering Task Force, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices of the Open Source Revolution', O'Reilly, Sebastapol, CA.

Dibell, J. (2004), 'We Pledge Allegiance to the Penguin', *Wired* **12**(11). Wired http://www.wired.com/wired/archive/12.11/linux.html [28 Dec 2004].

Galloway, A. R. (2004), *Protocol: How Control Exists after Decentralization*, MIT Press, Cambridge, MA.

Gillmor, D. (2004), *We the Media : Grassroots Journalism by the People, for the People*, O'Reilly, Beijing, Sebastopol, CA. 1st ed.

Hafstein, V. (2004), 'The Politics of Origins: Collective Creation Revisited', *Journal of American Folklore* **117**.

Himanen, P. (2001), *The Hacker Ethic, and the Spirit of the Information Age*, Random House, New York, NY. 1st ed.

Kaiser, J. (2004), 'Zerhouni Plans a Nudge toward Open Access', Science Magazine, 3 Sep 2004.

Kelty, C. M. (2004), 'Culture's Open Sources: Software, Copyright, and Cultural Critique', *Anthropological Quarterly* **77**(3).

Lessig, L. (2001), *The Future of Ideas : The Fate of the Commons in a Connected World*, Random House, New York, NY.

Lessig, L. (2004), *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, The Penguin Press, New York, NY. http://www.free-culture.cc/freeculture.pdf.

Raymond, E. S. (1999), A Brief History of Hackerdom, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices of the Open Source Revolution', O'Reilly, Sebastapol, CA.

Raymond, E. S. (2001), *The Cathedral and the Bazaar : Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, Beijing; Sebastopol, CA. Rev. ed.

Rosen, R. (2004), 'Top Ten Ideas of '04: Open Source Journalism, or „My Readers Know More Than I Do."', New York University, http://journalism.nyu.edu/pubzone/weblogs/pressthink/2004/12/28/tptn04_opsc.html [28 Dec 2004].

Stallman, R. (1999), The GNU Operating System and the Free Software Movement, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices of the Open Source Revolution', O'Reilly, Sebastapol, CA.

The Economist (2004), 'An Open-Source Shot in the Arm?', *The Economist* .

Torvalds, Linus et al. (2003), 'Revolution Os [Hackers, Programmers & Rebels Unite]', [S.l.] Los Angeles, Calif.: Wonderview Productions, Distributed by Seventh Art Releasing. videorecording.

Vaidhyanathan, S. (2001), *Copyrights and Copywrongs : The Rise of Intellectual Property and How It Threatens Creativity*, New York University Press, New York, NY.

Vaidhyanathan, S. (2004), *The Anarchist in the Library : How the Clash between Freedom and Control Is Hacking the Real World and Crashing the System*, Basic Books, New York, NY.

Wayner, P. (2000), *Free for All : How Linux and the Free Software Movement Undercut the High-Tech Titans*, HarperBusiness, New York, NY. 1st ed.

Weber, S. (2004), *The Success of Open Source*, Harvard University Press, Cambridge, MA.

Williams, S. (2002), *Free as in Freedom : Richard Stallman's Crusade for Free Software*, O'Reilly, Beijing; Sebastopol, CA.

Industrial Influences

Tile von Damm, Jens Herrmann und Jan Schallaböck

Seit Jahren zeigt sich die Musikindustrie in der Krise. Dass diese ausschließlich durch File-Sharing und CD-Brennen begründet ist, ist jedoch nicht zutreffend. Dennoch ermöglicht die singuläre Begründung unter dem Deckmantel der Kulturpolitik und der Innovationsförderung eine eilige, industriepolitisch geprägte Gestaltung des Rechtes, der Technik, ihrer Infrastrukturen und der Gesellschaft. Musik-, Film- und Computerindustrie betreiben derzeit, unterstützt durch den Gesetzgeber, mit Hochdruck die Einführung und Verbreitung von Digital Rights Management Systemen. Die Folgen dieser Entwicklung, in ihrer sich nun konkretisierenden Form, sind bislang noch wenig untersucht. Zumal die in Gang gesetzte Entwicklung weit über den Musik- und Filmbereich hinausgeht. Die Entwicklung von Alternativen, die geeignet sind, die vielfältigen Chancen der Digitalisierung konstruktiv umzusetzen, spielt nicht die Rolle, die ihr zukommen sollte. Die Autoren identifizieren aus technischem, politik- und rechtswissenschaftlichem Blickwinkel Schieflagen des derzeitig stattfindenden Prozesses. Nicht nur droht durch das neue Urheberrecht die Kriminalisierung Vieler, auch offenbaren sich ernst zu nehmende Schwierigkeiten für die Entwicklung und Verbreitung von Open–Source-Software.

1. Intro

Die Frage der digitalen Verfügbarkeit von Musik wird derzeit hoch kontrovers diskutiert. Die Vorschläge der Bundesregierung zur Regelung des Urheberrechts in der Informationsgesellschaft (Zweiter Korb) sollen die Rahmenbedingungen für die Online-Verwertung von Musik schaffen. Dass es jedoch in dieser Debatte um weit mehr als nur die Gestaltung des Musikbereiches geht, wird oftmals nicht wahrgenommen.

Die Musikindustrie versucht, weitreichende Auswirkungen auf den Umgang mit dem Internet, den Betriebssystemen und dem PC überhaupt zu erwirken, denn sie schickt sich an, die Tür für die Einführung von Digital Rights Management Systemen (DRM) zu öffnen.

Dies bleibt nicht ohne Auswirkung auf die Verbreitung von freie Software. Denn offene Standards kollidieren mit den derzeit favorisierten proprietären Lösungen.

Die Spezifika von Musik und die Struktur der Industrie begünstigen die Einführung von DRM. Alternativen, wie sie im Bereich freie Software vorliegen, genießen noch nicht die gesellschaftliche Relevanz, um die Einführung von DRM abzuwenden. Dies gilt ebenso für die (wenigen) vorliegenden Alternativen im Bereich freier Musik. Gerade im Bereich der Musik fehlt es an der vollständigen Übertragbarkeit der Freiheiten von Open Source.

2. Beat dis[1]

Die Merkmale der Lizenzen von freie Software lassen sich mit vier „Freiheiten" zusammenfassen:

1. Die Freiheit, ein Programm zu jedem beliebigen Zweck auszuführen;
2. Die Freiheit, die Funktionsweise eines Programms zu untersuchen und es an seine Bedürfnisse anzupassen;
3. Die Freiheit, Kopien weiterzugeben und damit seinem Nachbarn zu helfen;
4. Die Freiheit, ein Programm zu verbessern und die Verbesserung an die Öffentlichkeit weiterzugeben, so dass die gesamte Gesellschaft profitiert (Free Software Foundation 2004).

Seitdem Musik auch in digitaler Form gespeichert und verbreitet werden kann, gibt es Überlegungen, das Konzept von freie Software auch auf Musik anzuwenden.[2] Bislang am erfolgreichsten ist dies der Creative-Commons-Initiative (CC) gelungen.[3] Diese hat ein System von Lizenzen entwickelt, das derzeit an die Rechtssysteme einer Vielzahl von Ländern angepasst wird.[4] Im Gegensatz zur GNU General Public License[5] ist es nicht primär auf Software, sondern auf Texte, Bilder und eben auch auf Musik zugeschnitten. Gleichzeitig versucht die CC-Initiative, wesentliche Grundgedanken freier Softwarelizenzen aufzugreifen.

Die nach den freien Softwarelizenzen erforderliche Offenlegung des Quellcodes ist auf Musik nur schwer übertragbar. Dies gilt insbesondere bei analog produzierter Musik, denn hier gibt es nicht einen standardisierten Quellcode. Zudem lässt sich die menschliche Rezeption, Komposition und Interpretation eines Musikstückes nicht vollständig in Bits und Bytes beschreiben. Bei Open-Source-Software erfüllt die Offenlegung zwei zentrale Funktionen: Zum einen wird die Weiterentwicklung

1 Die Single „Beat Dis" von Bomb The Bass war eine der ersten kommerziell erfolgreichen, ausschließlich gesampelten Songs (Bomb The Bass 1988).

2 Ein früher Versuch war die Aktion „Open Music" vom Linuxtag 2001, die explizit auf freie Software rekurrierte (http://openmusic.linuxtag.org/).

3 Die nicht-kommerzielle Organisation Creative Commons gründete sich 2001, um dem System des „automatically copyrighted" freie und selbstbestimmte Lizenzen entgegenzusetzen (http://creativecommons.org/).

4 http://www.icommons.org/

5 Die GNU General Public License wird bei einer Vielzahl freier Softwareentwicklungen verwendet, unter anderem auch beim freien Betriebssystem Linux.

erleichtert. Zum anderen wird durch Transparenz ermöglicht, die Funktionsweise eines Programms zu untersuchen. Für letzteres ist der Bedarf bei Musik nicht gerade offenkundig. Entsprechend sehen auch die CC-Lizenzen keine Entsprechung zu den freien Softwarelizenzen vor.

Im Gegensatz zu Software ist ein Musikstück ein in sich geschlossenes, originäres Kunstwerk, bei dem fraglich ist, ob es einer Verbesserung in einem objektiven Sinne zugänglich ist. Durch die Bearbeitung eines Musikstückes entsteht etwas Neues, das neben dem Alten steht. Mit Sampling und Remixen sind jedoch neue Kunstformen entstanden, die besonders stark darauf aufbauen, Teile aus bestehenden Werken neu zusammenzufügen. Hierfür bietet Creative Commons eigene Sampling-Lizenzen an, die ausdrücklich die auszugsweise Weitergabe und Verwendung von Musik zu diesem Zweck gestattet.[6]

Grundelement der CC-Lizenzen ist die Nichtkommerzialität (Creative Commons 2004). Zwar darf man ein Musikstück neben einer CC-Lizenz auch kommerziell lizenzieren, die Frage, wie man mit einer Musikkomposition online (neben dem nicht abgedeckten Offline-Bereich) Geld verdienen kann, will die CC-Initiative aber grade nicht beantworten. Ihr Anliegen ist eine „freie Kultur", die auf eine geringe rechtliche und technische Regulierung digitaler Kulturgüter setzt (Lessig 2004). Will ein Rechteinhaber oder Rechteverwerter sich diesem Kulturbegriff nicht anschließen, kommen CC-Lizenzen für ihn nicht in Betracht.

Derzeit wird versucht, einen alternativen Weg zur Vergütung der Urheber zu finden. Neben dem klassischen kommerziellen Vertrieb liegt mit der „Berliner Erklärung zu kollektivverwalteten Online-Rechten" die Idee einer Pauschalabgabe zur Bezahlung des Rechtes auf File-Sharing vor (die sogenannte „Content- oder Kultur-Flatrate"[7]). Eine Online-Verwertungsgesellschaft soll danach Urheber und Verlage entsprechend der gemessenen Nutzung ihrer Werke vergüten.[8] Von staatlicher Seite müsste der rechtliche Rahmen hierfür geschaffen werden. Dieses Modell kollidiert jedoch mit den Vertriebs- und Verwertungsinteressen der großen Tonträgerhersteller, die eine Individualvergütung präferieren (Gebhardt 2004). Es ist daher nicht damit zu rechnen, dass sich dieses Modell durchsetzen wird.

6 Das ursprüngliche Gesamtstück darf nach diesen Lizenzen allerdings nicht weitergegeben werden.

7 (Berlin Declaration 2004)

8 Das Modell der Content-Flatrate ist bislang nur rudimentär ausgearbeitet. Unklar ist insbesondere, ob die rechtliche Umsetzung im Rahmen einer fakultativen oder einer obligatorischen Regelung erfolgen soll. Eine obligatorische (d. h. verpflichtende) Regelung wäre wohl verfassungswidrig und eine fakultative (freiwillige) weitgehend nutzlos. Weiterhin bleibt offen, ob die Urheber finanziell entsprechend der kommerziellen Verwertung entlohnt werden können. Daneben ist die Argumentation der Stärkung der Urheber gegenüber den Verwertern sicherlich nicht von der Hand zu weisen, jedoch ist anzumerken, dass nur ein kleiner Teil der musikalischen Aufnahmen heute tatsächlich Geld akquiriert. Eine Flatrate wird – zumindest im Offline-Bereich – dieses Modell sprengen. Lediglich zwischen 10 % und 15 % aller veröffentlichten Tonträger sind erfolgreich, d. h. spielen entweder Gewinn ein oder sind zumindest kostenrentabel (Wicke 1997, Becker und Ziegler 2000).

3. Pump Up The Volume[9]

Spätestens Anfang 2000, als die Nutzung von Internettauschbörsen und die Verbreitung von CD-Brennern stark anstieg, sah sich die Musikindustrie zum Handeln gezwungen. Die bis heute verfolgte Strategie der Tonträgerindustrie besteht aus zwei wesentlichen Teilen: Zum einen die Entwicklung und Implementierung technischer Schutzsysteme und zum anderen der Ausweitung der Rechtslage auf die „Bedürfnisse der Mediengesellschaft".

Seit vier Jahren ist der weltweite Umsatz innerhalb der Tonträgerbranche rückläufig. Die offiziellen Zahlen des Bundesverbandes der Phonographischen Wirtschaft[10] attestieren einen Rückgang in Deutschland von real 20,9 % (2003) im Vergleich zum Vorjahr (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 8 f.). Die beiden Hauptgründe sind der Branche zufolge das Brennen von CDs sowie die digitale Verbreitung über das Internet, insbesondere über Tauschbörsen.

Als angeblichen Beweis stellt sie die Zahlen der verkauften CD-Rohlinge den verkauften Tonträgern gegenüber. Hierbei führt sie nicht verifizierbare Zahlen für kopierte CDs (350 Millionen) und Downloads aus „illegalen" Quellen (600 Millionen Songs) an (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 7 f.). Dennoch liegt der hochgerechnete Gesamtumsatz der Musikbranche in Deutschland 2003 bei 1,816 Milliarden Euro (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 8 f.). Das entspricht noch immer dem Niveau von 1990 und ist sogar doppelt so hoch wie 1985 (Kuri 2003*a*). Kann man also wirklich von einer Krise sprechen?

Der rasanten technischen Entwicklung hat die Musikindustrie keine konstruktive Lösung entgegengesetzt. Die „Napsterisierung" wurde zum Synonym für die beginnende Krise der Tonträgerbranche. Dennoch ist es nicht korrekt, den Umsatzrückgang der Musikindustrie ausschließlich auf die technischen Veränderungen und Möglichkeiten zurückzuführen. Die Gründe für die Krise liegen vielmehr in der Kulmination verschiedener Faktoren, die die Musikindustrie nur am Rande anerkannt hat: Die Musikindustrie ist im hohen Maße von der Rezession abhängig. Als Unterhaltungsindustrie lebt sie vom Freizeit- und Konsumverhalten ihrer Käufer/innen (Harker 1998). Dabei konkurriert der Tonträger mit anderen Medienprodukten und Freizeitaktivitäten. Teilweise substituieren diese den Tonträger bzw. führen zu einem Kaufkraftabfluss (Kulle 1998, S. 197). Die anhaltende wirtschaftlich gespannte Lage in Deutschland ist also ein wesentlicher Faktor für die rückläufigen Verkaufszahlen.

Die Umsätze der Musikindustrie bewegten sich Mitte der 1990er Jahre auf einem bis heute einmalig hohen Niveau. Dies ist vor allem der erfolgreichen Einführung der Compact Disc und der parallelen massiven Ersetzung der Vinylplatte geschuldet. Ein erfolgreiches neues Format konnte seitdem – trotz mehrerer Versuche – nicht

9 Die auf dem kleinen Independentlabel 4AD veröffentlichte Single „Pump Up The Volume" erreichte 1987 die #1 der UK-Charts. Der Song ist einer der maßgeblichen Wegbereiter des Acid House und der Sampling-Culture (M/A/R/R/S 1987).

10 Der Bundesverband der Phonographischen Wirtschaft ist die deutsche Landesgruppe der International Federation of the Phonographic Industry (IFPI). Er repräsentiert in Deutschland 91 % des deutschen Musikmarktes.

eingeführt werden.[11] Zudem konzentriert sich der Markt zu über 70 % auf die fünf größten Plattenfirmen (Becker und Ziegler 2000, S. 20).[12] Ihre starke Stellung wird zusätzlich dadurch begünstigt, dass ein Großteil der Independentlabels über Vertriebs- und Beteiligungsabkommen mittlerweile an die großen Firmen gebunden ist.[13]

Trotz der bis Ende der 1990er steigenden Verkaufszahlen, sah sich die Branche der sinkenden Rentabilität ihrer Produkte ausgesetzt, der sie kein Konzept entgegengesetzt hat. Dies liegt zum einen an der abnehmenden klassischen Aufbauarbeit und langfristigen Förderung von Künstlern zugunsten schneller, Gewinn versprechender Produkte. Beim Abebben eines Trends sind durch die ausbleibende weitere Förderung eines Künstlers die Ausgangsinvestitionen für Produktion und Marketing verloren (Friedrichsen 2004, S. 37). Der daraus resultierende „wirtschaftliche Zwang" führt zu einer steigenden Anzahl der Veröffentlichungen und einem Steigen des Break-Even Punktes.[14] Hinzu kommt, dass – insbesondere für die erfolgreichen Künstler – nicht nur die Kosten für Marketing und Promotion, sondern auch die erfolgsunabhängige Vergütung sowie teilweise die Produktionskosten sprunghaft angestiegen sind.[15]

Die Gründe für die Krise der Tonträgerindustrie sind also vielfältig und hauptsächlich innerhalb der Branche zu suchen. Neben den internen Faktoren haben allerdings die externen Faktoren wesentlich zu einer Beschleunigung der Krise beigetragen. Neben der wirtschaftlichen Rezession sind sicherlich die technischen Möglichkeiten – vor allem das File-Sharing und das CD-Brennen – zu nennen.

4. Blue Monday[16]

Eine wesentliche Stärke der Tonträgerindustrie lag bislang in der Herstellung und dem Vertrieb eines Tonträgers, der aber bei der Distribution von Musik über das

11 Zu nennen ist hier bspw. die von Sony entwickelte Minidisc (MD). Einzig die DVD erreichte in den vergangenen Jahren nennenswerte, stetig steigende Verkäufe, so dass die Phonoindustrie ihr bereits bis 2010 einen Absatz in der Größenordnung der CD prophezeit (Gebhardt 2003).

12 Mit der Fusion von *Sony Music* und der *Bertelsmann Music Group* (BMG) entsteht nach Universal Music die gemessen am Umsatz weltweit zweitgrößte Plattenfirma. Die Anfang August 2004 vollzogene Fusion wird Anfang 2005 umgesetzt (Netzeitung 2004). Neben Sony BMG Music Entertainment bilden somit Universal Music, die Warner Music Group und EMI Music die größten Tonträgerunternehmen, die so genannten Majors.

13 Die Bezeichnung „Independents" oder kurz „Indie" stand ursprünglich tatsächlich für einen von den großen Firmen unabhängigen wirtschaftlichen Status. Seit den 1970er Jahren allerdings ist dies in den meisten Fällen so nicht mehr gültig.

14 Der Break-Even Punkt ist die Grenze, an der eine Produktion Gewinn einspielt. Eine weitere Entwertung der Musik ergibt sich durch die steigende Zahl der formatierten Sendeformen im Rundfunk. Die gesendete Musik reduziert sich auf den kleinsten gemeinsamen Nenner und wird auf seine „Abschaltfestigkeit" hin geprüft. Zunehmend jedoch orientierte sich die Tonträgerindustrie in ihrer Veröffentlichungspolitik am Sendeformat.

15 Ein absurdes Beispiel hierfür ist sicherlich die Auflösung des laufenden Vertrags mit Mariah Carey seitens des Labels Virgin/EMI, nachdem ihr Album „Glitter" mit „nur" vier Millionen verkaufter Exemplare als Flop angesehen wurde. Sie bekam eine Abfindung von 32 Millionen US-Dollar (Lau 2002, S. 51 f.).

16 Oder „How Does It Feel To Treat Me Like You Do". Die Elektronikpioniere New Order veröffentlichten 1983 ausschließlich auf 12" den Klassiker „Blue Monday" (New Order 1983).

Internet nicht zwangsläufig notwendig ist. Vielmehr kann jeder abgekoppelt von einem physischen Tonträger seine Musik über das Internet vertreiben. Das zentrale Geschäftsmodell der Musikindustrie – insbesondere der Majors – ist also bedroht.

Neue kommerzielle Vertriebs- und Verwertungskonzepte, die finanziellen Erfolg versprechen, entstehen nur zögerlich. Zwar haben sich die Majors inzwischen zu globalen Medienunternehmen entwickelt,[17] die neben dem klassischen Tonträgerbereich eben auch digitale Vertriebswege bereithalten. Die Uneinigkeit der Branche, eine fehlende technische und rechtliche Ausgestaltung und die (finanziell) nach wie vor erfolgreiche klassische Verwertung haben bislang keine vernünftigen Alternativvertriebswege ermöglicht. Dieses zeigt auch der lange Weg zu einer gemeinsamen Online-Plattform der Musikindustrie (Kuri 2004). Worüber sich die Tonträgerbranche allerdings einig ist, ist die Forderung, dass beim Musikvertrieb über das Internet die vertriebenen Daten vor unberechtigter Vervielfältigung geschützt werden sollen.

Digital Rights Management (DRM) ist die Technologie, mit deren Hilfe die Durchsetzung von Nutzungslizenzen für digitale Daten ermöglicht werden soll. Mit Hilfe von DRM soll eine der Grundeigenschaften digitaler Daten verhindert werden, nämlich die Möglichkeit, nahezu ohne Aufwand verlustfreie Kopien herzustellen, damit sich auch diese Daten wie Verbrauchsgüter vermarkten lassen.

DRM soll ein „umfassendes Vertriebskonzept für digitale Güter" ermöglichen (Günnewig 2002). Neben der reinen Nutzungs- und Zugangskontrolle (Kopierschutz)[18] sollen DRM-Systeme auch Authentizität und Integrität von Daten sicherstellen, Metadaten zu den übertragenen Informationen liefern und die komplette Verwertungskette vom Bereitstellen der Daten bis zur Übertragung zum Kunden und der Abrechnung verwalten (Fetscherin 2004).

Kritiker nennen DRM auch „Digital Restrictions Management", also Beschränkungsverwaltung (Stallman 2002). Da es um die Verhinderung bestimmter Nutzungen digitaler Daten geht, ist dieser Begriff durchaus zutreffend.[19]

Der bislang erfolgreichste Versuch, Musik über das Internet zu vertreiben, kommt von der Firma *Apple* (Herrmannstorfer 2004). Auf der Apple eigenen Online-Verkaufsplattform erworbene Musik kann man (ohne vorherige Konvertierung) nur auf die von Apple entwickelten und lizenzierten Abspielgeräte kopieren. Auch Audio–CDs lassen sich nur in beschränkter Anzahl herstellen. Dazu wird beim Installieren der Abspielsoftware, die ebenfalls von Apple kommt, auf einem Computer ein für diesen eindeutiger „machine identifier" erzeugt, der diese Beschränkung sicherstellt, wobei das DRM nicht in Abhängigkeit vom Dokument variiert werden kann.

17 Beispielsweise fusionierten 2001 Time Warner und AOL zu dem weltweit größten Medienkonzern.

18 So bei „Lightweight DRM" (Frauenhofer 2003). Hierbei werden die Daten nicht verschlüsselt und technisch kopiergeschützt, sondern lediglich mit einem eindeutigen Wasserzeichen versehen. Anhand dieses Wasserzeichens kann ein Kunde ermittelt werden, der Rechte an einer Datei erworben und die Datei ggf. unrechtmäßig verbreitet hat. Hier entsteht der Schutz vor unrechtmäßiger Vervielfältigung nicht direkt durch technische Maßnahmen. Wie im klassischen Urheberrecht ist der Urheber zur Durchsetzung seiner Interessen auf staatliche Maßnahmen (also Zivilverfahren und Strafverfolgung) angewiesen.

19 Insoweit technisch unterbunden wird, was rechtlich erlaubt ist, ist offenkundig auch die Bezeichnung „Rights Management" irreführend, denn es wird ja gerade nicht das Recht verwaltet.

Microsoft bietet ebenfalls ein DRM-System an, das mit dem Windows eigenen Mediaplayer zusammenarbeitet (Microsoft Deutschland 2004). Dieses System wird auf diversen Download-Plattformen für Musik verwendet (Hansen und Block 2004, S. 176 f.). Im Gegensatz zur Lösung von Apple kann der *Windows Media Rights Manager* beliebige unterschiedliche Lizenzen verwalten.

5. Sour Times[20]

Die beschriebenen Systeme teilen eine gemeinsame Schwachstelle: Die Abspielsoftware auf dem Computer des Nutzers muss in das DRM-System einbezogen sein. Sie entschlüsselt die Daten und gibt sie aus, und sie „entscheidet" bei vorliegenden Schlüsseln auch anhand der Lizenz, ob ein bestimmtes Stück überhaupt bzw. wie oft es abgespielt werden darf.

Dabei besteht die Gefahr, dass die Daten bei der Abspielsoftware in entschlüsselter Form „abgegriffen" werden. Der Schutz, den das DRM-System gegen unrechtmäßiges Verbreiten der Daten bieten soll, ist somit leicht zu umgehen.

Um unter anderem auch diesem Problem Herr zu werden, gründete sich die *Trusted Computing Group*[21], ein Industriekonsortium, das versucht, ein System zu entwickeln, mit dessen Hilfe es möglich sein soll, auf dem Rechner eines Anwenders nur bestimmte, als sicher bekannte Software zu verwenden. Dabei soll die Hardware des PC um ein Modul (ähnlich einer Smartcard) erweitert werden, mit dem kryptographische Schlüssel gespeichert werden. Diese sind für den Eigentümer des PC unzugänglich, ermöglichen aber einem Kommunikationspartner, also auch dem Online-Anbieter von Musik, sicherzustellen, dass nur im Rahmen der technischen Vorgaben des Anbieters auf die Daten zugegriffen werden kann.

Dabei besteht die Gefahr, dass durch ein proprietäres *Trusted Computing* (TC) die Entwicklung von Open-Source-Software bis zur Unmöglichkeit erschwert werden kann (Stallman 2002). Da jede Anwendung, die in einem sicheren Umfeld ausgeführt werden soll, zertifiziert und signiert werden muss, entstehen für Open Source neue Kosten, die die Entwicklung von Open-Source-Software zumindest für einzelne freie Entwickler drastisch erschwert. Aber auch für reine Windows-Anwender entstehen durch ein proprietäres TC Probleme. Bei einer intransparenten Zertifizierungsstruktur wird dem Nutzer die Selbstbestimmung über seinen eigenen Computer entzogen (Anderson 2004). Es sind offene Standards notwendig, um einen selbstbestimmten Zugang zu digitalen Medien zu gewährleisten.

Gleichzeitig kann eine Technologie wie TC helfen, ein Problem im Zusammenhang von Open Source und DRM zu lösen. DRM dient grundsätzlich dazu sicherzustellen, dass entschlüsselte Daten nur unter den Bedingungen der dazugehörigen Lizenz zugänglich sind. Ein DRM-System basierend auf Open Source muss auch die Routine

20 Mit ihrer 1994 veröffentlichten zweiten Single „Sour Times" und dem Album „Dummy" zählen Portishead zu den Pionieren des TripHop (Portishead 1994).

21 Die *Trusted Computing Group* (TCG) versteht sich selber als „not-for-profit organization". Als „Promoter" fungieren: AMD, HP, IBM, Intel, Microsoft, Sony und Sun (https://www.trustedcomputinggroup.org/).

enthalten, die die verschlüsselten Daten entschlüsselt und weiterverarbeitet. Da der Code offen liegt, kann diese Routine dann dazu benutzt werden, die entschlüsselten Daten aus dem DRM-System zu befreien. Der Schutz ist damit hinfällig. Zwar ist immer noch ein Schlüssel erforderlich, um die Daten zu entschlüsseln, was danach mit ihnen geschieht, liegt allerdings nicht mehr in der Hand derjenigen, die die Verschlüsselung vorgenommen haben.

Anliegen von DRM ist es allerdings, genau dies zu unterbinden. Die klassische Antwort auf dieses Problem ist Closed-Source-Software. Eine andere Lösung wäre es sicherzustellen, dass nur Code ausgeführt wird, der als sicher bekannt (und signiert) ist. Ein solcher Schutz muss dann allerdings durch die Hardware sichergestellt werden. Damit könnten ohne Weiteres die Quellen der Software veröffentlicht werden. Sie können nur nicht modifiziert werden, ohne erneut signiert zu werden. *Trusted Computing* könnte also Open-Source-DRM-Systeme ermöglichen.

6. What time is love?[22]

Aus einem tradierten, überkommenen Verständnis von „geistigem Eigentum", das auf John Locke zurückgeführt wird, hat der Urheber ein „angeborenes Recht" an den von ihm geschaffenen Gütern (Kirchhof 1998, S. 2). Aus dieser vorherrschenden Sicht auf das Urheberrecht ist die Einführung von DRM durchaus konsequent, schließlich ermöglicht es dem Urheber in umfänglicher Weise, darauf Einfluss zu nehmen, was mit seinem Werk geschieht.

Meist wird mittlerweile das Urheberrecht auf die Anspornttheorie gestützt (Oechsler 1998). Das Urheberrecht soll Anreize für geistige Tätigkeit und Schöpfung schaffen. Dies entspricht dem Bild einer auf Fortschritt ausgerichteten Gesellschaft. Der Schutz, den das Urheberrecht dem Schöpfer gewährt, mache es für ihn attraktiver, sein Werk zu veröffentlichen, und gäbe der Gesellschaft die Möglichkeit, von der Leistung des Schöpfers zu profitieren.

Mit dem *Gesetz zur Reform des Urheberrechtes in der Informationsgesellschaft*, dessen „Zweiter Korb" nunmehr als Referentenentwurf vorliegt, geht der Gesetzgeber noch einen Schritt weiter; das Gesetz „bemüht sich . . .um einen Interessensausgleich" (BMJ 2004, S. 19). Dem liegt ein Urheberrechtsverständnis zugrunde, das stark auf Vertragsrecht rekurriert. Wegen der Sozialbindung des (geistigen) Eigentums[23] hat der Gesetzgeber, ähnlich wie etwa im Verbraucherschutz sowie im Arbeits- oder Mietrecht, die Aufgabe, den Schwachen vor dem Starken zu schützen.

DRM bringt eine neue Problemstellung, für die der Gesetzgeber neue Wege finden muss. Vor der Einführung von Tonträgern Anfang des 20. Jahrhunderts war es schon technisch unmöglich, Kopien herzustellen. Ein Recht auf Privatkopie wäre damals

22 „The KLF – What Time Is Love" erschien 1989 auf dem eigenen Label KLF Communications. Das Avantgarde-Projekt „The Justified Ancients of Mumu", alias The KLF (Kopyright Liberation Front) erreichten unter dem Projektnamen The Timelords 1988 mit „Doctorin' The Tardis" die #1 der UK-Charts (KLF 1989, Drummond und Cauty 1988).

23 Art. 14 GG.

also völlig nutzlos gewesen. Technische Neuerungen erweiterten die Handlungsmöglichkeiten der Verbraucher. Mit der Verbreitung von Tonbändern und Audiokassetten wurde es möglich, ohne großen Aufwand Musik privat zu kopieren und weiterzugeben.[24] Mittels DRM werden die neuen Möglichkeiten des Kopierens und Verbreitens von urheberrechtlich geschütztem Material technisch unterbunden. Neu ist also, dass der Verbraucher in seinen Handlungsmöglichkeiten eingeschränkt wird.

Grundsätzlich gilt dabei die allgemeine Handlungsfreiheit[25] auch für den Urheber. Wenn jemand mittels DRM die technischen Möglichkeiten beschränken will, kann er das zunächst tun, ein Verbot von DRM würde einen Grundrechtseingriff darstellen. Im Datenschutz, bei personenbezogenen Daten, wird man dem „Inhaber" das Recht, „seine" Daten zu verschlüsseln, nicht verwehren wollen.[26] Genauso dürfte man sie wohl auch mit DRM ausstatten und so die Entschlüsselung nur für eingeschränkte Zwecke zulassen. Ein Grundrechtseingriff kann aber beispielsweise gestattet sein, wenn man sich in den Geschäftsverkehr begibt und seine Musik verkaufen will. Hier könnte man unter anderem wegen der Sozialbindung des geistigen Eigentums oder der Wissenschaftsfreiheit Eingriffe in die allgemeine Handlungsfreiheit des Urhebers rechtfertigen.

Dieser Weg wird aber mit dem vorliegenden Referentenentwurf nur zögerlich beschritten. Stattdessen stützt er die Geschäftsmodelle, mit denen die Tonträgerindustrie ihre Marktstellung behaupten möchte. So wird versucht, den Down- und Upload in File-Sharing-Börsen unter Strafe zu stellen. Zwar verkündete die Bundesjustizministerin vollmundig, es gehe nicht um die „Kriminalisierung der Schulhöfe" (Stein 2004, S. 4). Doch die angekündigte Bagatellausnahme, die das Downloaden einzelner Songs aus illegalen Tauschbörsen[27] von Strafe ausnimmt, geht an der Realität der Schulhöfe und auch weitaus größerer Bevölkerungskreise vorbei. Denn es werden in aller Regel nicht einzelne Songs aus illegalen Tauschbörsen, sondern hunderte Songs in legalen Netzwerken getauscht (Kuri 2003*b*).

Das Justizministerium nimmt, um der Musikindustrie ihre Vorstellungen von der Online-Verwertung zu sichern, die Kriminalisierung von Millionen von File-Sharing–Nutzern sehenden Auges[28] in Kauf. Dies ist rechtspolitisch wenig wünschenswert. Es besteht eine Diskrepanz zwischen den Nutzungsgewohnheiten Vieler und der

24 Diese Möglichkeit wurde vom Bundesverfassungsgericht legitimiert mit der Begründung, dass das Kopieren nicht zu unterbinden sei ohne weitreichende, nicht legitimierte Eingriffe in die Privatsphäre der Menschen.

25 Art. 2 GG, oder: „Der Mensch ist frei. Er darf tun und lassen, was die Rechte anderer nicht verletzt oder die verfassungsmäßige Ordnung des Gemeinwesens nicht beeinträchtigt" (Land Hessen 1946, Art. 2).

26 Interessanterweise war dies Gegenstand einer langen Debatte. Bei personenbezogenen Daten war lange Zeit die Anordnung einer Schlüsselhinterlegung im Gespräch (Computerwoche 1997).

27 Das Gesetz formuliert die Regelung freilich, ohne direkt auf Peer-to-Peer-Netzwerke Bezug zu nehmen und stellt auch klar, dass der Download auch im Rahmen der Bagatellausnahme widerrechtlich bleibt, so dass zivilrechtliche Ansprüche gegen den Tauschbörsennutzer nicht ausgeschlossen bleiben: „Nicht bestraft wird, wer rechtswidrig Vervielfältigungen nur in geringer Zahl und ausschließlich zum eigenen privaten Gebrauch herstellt" (BMJ 2004, §106, Abs. 1, Satz 2(neu)).

28 Das ist ein Ergebnis einer unveröffentlichten, für das Ministerium entwickelten Online-Befragung zu den Überzeugungen von Tauschbörsennutzern. Sie zeigte, dass etwa ein Drittel der etwa sieben Millionen derzeitigen Nutzer nicht mit Aufklärungskampagnen zu erreichen sind. (Kemmler 2004)

willkürlichen Verfolgung Einzelner.

Bei der Einführung von Tonbandgeräten jedenfalls hat man sich für einen anderen Weg entschieden und das Recht auf Privatkopie eingeführt, um mittels Pauschalabgaben und Verwertungsgesellschaften die Vergütung zu sichern.[29] Heute wird ein ähnliches System gefordert: Die Kultur-Flatrate.

Derzeit ist noch offen, ob es gelingt, mit dem rabiaten Mittel des Strafrechts den Tauschbörsen das Wasser abzugraben. Die Nutzungszahlen deuten nicht darauf hin, ein Rückgang ist bisher nicht feststellbar (Wilkens 2004), obwohl die Musikindustrie beginnt, Klagen gegen die Nutzer anzustrengen und die Filmindustrie mit einer provokanten Werbekampagne[30] versucht, auf die Nutzer einzuwirken.

Neben der Strafbestimmung für das Downloaden sieht das neue Urheberrecht ein Verbot der Umgehung technischer Schutzmaßnahmen vor[31] (BMJ 2004, S. 19). Auch auf diesem Wege begünstigt es die Einführung von DRM-Systemen. Liegt ein Schutzmechanismus vor, werden digitale Beschränkungen des Urheberrechts Makulatur, d. h. der Nutzer darf seine Rechte nicht mehr ausüben, wenn ein technischer Schutzmechanismus vorliegt.[32] Ein Berechtigung zu digitalen Kopien ist nicht durchsetzbar, wenn Kopierschutzmaßnahmen bestehen. Der Gesetzgeber ermutigt die Musikindustrie de facto, die Werke mit Kopierschutz- oder DRM-Systemen zu versehen, da sie sonst die Privilegisierung des neuen Urheberrechts nicht für sich in Anspruch nehmen kann. Zwar werden CDs zunehmend wieder ohne Kopierschutz ausgeliefert, weil die Kunden die mit den kopiergeschützten „Un-CDs“[33] einhergehenden Beschränkungen nicht akzeptieren. Beim Onlinevertrieb zeichnet sich aber ab, dass sich DRM-Verfahren durchsetzen, die die Kopierbarkeit der Daten beschränken.

Der Gesetzgeber stützt also die Einführung von DRM im Rahmen der Reform des Urheberrechtes. Ansätze für eine Regulierung von DRM finden sich aber nicht. Selbst die Diskussion um die Auswirkungen von DRM findet bisher nur sehr zaghaft statt. Lediglich eine Expertenrunde der Europäischen Kommission, die *High Level Group* (HLG) des Aktionsplans „eEurope 2005“, beschreibt die Erfordernisse bei einer breiten Einführung von DRM (High Level Group 2004). Vor allem die erhobene Forderung nach Interoperabilität von DRM auf unterschiedlichen Systemen ist im Zusammenhang mit Open Source von immenser Bedeutung.[34]

---

29 Vgl. BGHZ 42, 118; BVerfG GRUR 1980, 44, 48.

30 Gegenüber dieser hat selbst der Zentralverband der deutschen Werbeindustrie Bedenken geäußert hat (Kuri 2003*c*).

31 Hierbei setzt die Bundesregierung völkerrechtliche Verpflichtungen um, die sie bereits 1996 eingegangen ist (WIPO 1996).

32 Unter die so genannten Urheberrechtsschranken fallen unter anderem das Recht zur Kopie für Zwecke der Wissenschaft und Lehre, das Zitatrecht in beschränktem Umfang, die Privatkopie, die Vervielfältigung für gerichtliche und behördliche Verfahren und andere.

33 Der Begriff „Un-CD“ geht darauf zurück, dass die derzeitig zumeist eingesetzten Kopierschutzverfahren bei CDs mit einer Abweichung vom allgemein vereinbarten CD-Standard ermöglicht werden, was auch dazu führt, dass sie sich auf vielen CD-Playern nicht mehr abspielen lassen.

34 An der HLG wurde bemängelt, dass hier die Interessen der Wirtschaft, insbesondere der Verwertungsindustrie deutlich im Vordergrund standen und die Bedürfnisse der Verbraucher vernachlässigt würden (Grassmuck 2004). So wurde nur eine einzige Verbraucherschutzorganisation, das *Bureau Européen des Unions de Consommateurs* (Europäische Verbraucherschutzorganisation, BEUC), zu den Verhandlun-

Die HLG präferiert eine Lösung auf der Basis offener Standards, damit keine Herstellerabhängigkeiten entstehen. Solche offenen Standards sind in Ansätzen bereits vorhanden, z. B. die *MPEG21 Rights Expression Language*, die eine Beschreibung von erteilten Nutzungslizenzen ermöglicht (Wang 2004). Allerdings sind vor allem Implementierungen erforderlich, um ihre Durchsetzung zu fördern.

Derzeit existieren lediglich „Inseln proprietärer Systeme" (High Level Group 2004), die untereinander inkompatibel sind. Aus diesen kann sich ein De-facto-Standard herausbilden. Dies geschieht in der Regel dann, wenn ein Anbieter sich am Markt durchgesetzt hat. Um Plattformabhängigkeiten zu vermeiden, müsste dieser die Spezifikation des Standards öffnen. Dies ist weder beim Angebot von Apple noch dem von Microsoft der Fall. Eine Portierung auf offene Betriebssysteme scheint unwahrscheinlich. Ein offenes Betriebssystem wie Linux wird also bei der Fortschreibung der derzeitigen Entwicklung keine Alternative zu den proprietären Systemen darstellen können.

7. Outro

Das Geschäftsmodell einer Industrie, die sich selbst noch als „Tonträgerindustrie" bezeichnet, hätte weit weniger Überlebenschancen, wäre diese Branche nicht so stark konzentriert. Diese Konzentration und ihre bereits existierende starke Rechtsposition ermöglicht ihr einen erheblichen Einfluss auf die Politik. Diesen Einfluss setzt sie für die weitere Stärkung ihrer Verwertungsrechte ein und nötigt dabei den Verbrauchern eine DRM-Infrastruktur auf. Die Gefahr besteht, dass es nun zu der Durchsetzung eines proprietären De-facto-Standards kommt, mit weitreichenden Konsequenzen auch auf anderen Gebieten als nur dem Musikvertrieb im Internet.[35] Bei einem proprietären Standard sind darüber hinaus auch freie Betriebssysteme außen vor, was ihre Chancen auf eine Verbreitung insbesondere unter Privatnutzern drastisch reduziert. Es stellt sich also die Frage, ob eine derartige Technologie allein der Unterhaltungsindustrie und ihrer Lobby überlassen werden soll. Es ist dringend geboten, die Agenda um Verbraucherperspektiven zu erweitern, um einer aggressiven Industriepolitik Einhalt zu gebieten.

Nur unter Einbeziehung von Open Source kann es gelingen, transparente Strukturen zu gewährleisten und Verbrauchern zu ermöglichen, statt Restriktionen die eigenen Rechte zu verwalten. Es droht eine einseitige rechtliche und technische Ausgestaltung von DRM. Perspektiven für Open Source im Bereich DRM sind schwer erkennbar. Allerdings kann ausgerechnet die Technologie, die bei *Trusted Computing* (TC) zum Einsatz kommt, hierfür eine Lösung bieten. Das Vertrauen, das bei proprietärer Software aus der Nichtanpassbarkeit resultiert, kann bei TC in die Zertifizierungsprozesse verlagert werden. Dabei muss es allerdings auf offenen Standards und standardisierten

gen eingeladen. Außerdem standen Verbraucheraspekte als letzter Punkt auf der Tagesordnung, konnten schließlich aus Zeitmangel nicht mehr behandelt werden. Sie tauchen im Abschlussreport deshalb nicht auf (Ermert und Kuri 2004).

35 Interessant erscheint in diesem Zusammenhang auch die Idee, Datenschutz mit Hilfe von DRM-Systemen zu ermöglichen (Tóth 2004).

Zertifikaten (wie etwa *common criteria*[36]) aufbauen. Denn es darf nicht übersehen werden, dass Abhängigkeiten von denjenigen entstehen, die die Zertifikate ausstellen. Wenn es möglich sein soll, auch mit Open-Source-Software DRM-geschützte Inhalte abzuspielen, müssen die Anbieter von DRM-Inhalten ihren DRM-Schutz so ausgestalten, dass jede Software, die den Anforderungen an den technischen Schutz genügt, auch ein Zertifikat erhalten kann.

Literaturverzeichnis

Anderson, R. (2004), 'Trusted Computing' Frequently Asked Questions – TC / TCG / LaGrande / NGSCB / Longhorn / Palladium / TCPA', http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html [29. Okt 2004].

Becker, A. und Ziegler, M. (2000), 'Wanted: Ein Überlebensmodell für die Musikindustrie Napster und die Folgen'. White Paper der Diebold Deutschland GmbH, Eschborn.

Berlin Declaration on Collectively Managed Online Rights: Compensation without Control (2004), http://www.wizards-of-os.org/index.php?id=1699 [18. Okt 2004].

Bomb The Bass (1988), 'Beat Dis', Mister-Ron DOOD1.

Bundesministerium der Justiz (BMJ) (2004), 'Referentenentwurf für ein zweites Gesetzt zur Regelung des Urheberrechtes in der Informationsgesellschaft'.

Bundesverband der Phonographischen Wirtschaft e. V. (2004), 'Jahreswirtschaftsbericht 2003'.

Computerwoche (1997), 'Siegen die liberalen Kräfte im Streit ums Kryptogesetz?', *Computerwoche* **14**(14), S. 9–10.

Creative Commons (2004), 'Frequently Asked Questions', http://creativecommons.org/faq [15. Sep 2004].

Drummond, B. und Cauty, J. (1988), *The Manual*, KLF Publications/Ellipsis, London.

Ermert, M. und Kuri, J. (2004), 'Europäische Kommission startet Konsultation zu DRM', heise online http://www.heise.de/newsticker/meldung/49253 [20. Okt 2004].

Fetscherin, M. (2004), 'Digital Rights Management Systeme: Stand der Technik'. http://www.ie.iwi.unibe.ch/staff/fetscherin/resource/2004-01-19-DRM-Stand-der-Technik.pdf [18. Sep 2004].

Fraunhofer-Institute for Integrated Circuits (IIS) / Fraunhofer-Institute for Digital Media Technology (IDMT) / Fraunhofer-Institute for Secure Telecooperation (SIT) (2003), 'Light Weight Digital Rights Management LWDRM®', http://www.lwdrm.com/ [10. Okt 2004].

Free Software Foundation (2004), 'Was ist Freie Software?', http://www.germany.fsfeurope.org/documents/freesoftware.de.html [10. Okt 2004].

Friedrichsen, M. e. A. (2004), *Die Zukunft der Musikindustrie – Alternatives Medienmanagement für das mp3-Zeitalter*, Verlag Reinhard Fischer, München.

36 http://www.commoncriteriaportal.org/

Gebhardt, G. (2003), 'Die Musikwirtschaft im Jahr 2010', http://www.ifpi.de/news/news-312.htm [20. Okt 2004]. Keynote auf der popkomm am 15.8.2003.

Gebhardt, G. (2004), 'Sieben Argumente gegen eine Kulturflatrate', http://www.spiegel.de/netzwelt/politik/0,1518,316837,00.html [10. Okt 2004].

Grassmuck, V. (2004), 'Putting users at the centre achieving an 'information society for all'', http://privatkopie.net/files/privatkopie-bof_on-DRM.pdf [20. Okt 2004].

Günnewig, D. (2002), 'Digital Rights Management Systeme: Ein Helfer in der Not?', http://www.datensicherheit.nrw.de/Daten/WS06122002/guennewig.pdf [30. Sep 2004].

Hansen, S. und Block, A. (2004), 'Musik saugen legal – Die Türen zu den Online-Shops sind aufgestoßen', *c't* **6**, S. 176–187.

Harker, D. (1998), 'It's a jungle sometimes. The music industry, the crises and the state'. http://www2.hu-berlin.de/fpm/texte/harker1.htm [01. Nov 2004].

Herrmannstorfer, M. (2004), 'iTunes Music Store als Vorbild für die Musikindustrie', heise online http://www.heise.de/newsticker/meldung/51741 [18. Okt 2004].

High Level Group (2004), 'High Level Group on Digital Rights Management', http://europa.eu.int/information_society/eeurope/2005/all_about/digital_rights_man/doc/040709_hlg_drm_2nd_meeting_final_report.pdf [20. Okt 2004]. Final Report.

KLF (1989), 'What Time Is Love', KLF Communications KLF004.

Kemmler, Sebastian, e. a. (2004), Das Urheberrecht in der Informationsgesellschaft. Studie zum Meinungsbild zur Novellierung des Urheberrechtsgesetzes. Studentische Projektgruppe Infrarot an der Universität der Künste, Berlin.

Kirchhof, P. (1998), *Die Zukunft der Musikindustrie – Alternatives Medienmanagement für das mp3-Zeitalter*, v. Decker und Müller, Heidelberg.

Kulle, J. (1998), Ökonomie der Musikindustrie: Eine Analyse der körperlichen und unkörperlichen Musikverwertung mit Hilfe von Tonträgern und Netzen, PhD thesis, Universität Hohenheim.

Kuri, J. (2003*a*), 'Musikindustrie sieht noch kein Ende der Talfahrt', heise online http://www.heise.de/newsticker/meldung/39996 [24. Aug 2004].

Kuri, J. (2003*b*), 'US-Gericht verweigert Schließung von Online-Tauschbörsen', heise online http://www.heise.de/newsticker/meldung/36411 [16. Nov 2004].

Kuri, J. (2003*c*), 'Werbeverband hält Kampagne gegen Raubkopierer für äußerst fragwürdig', heise online http://www.heise.de/newsticker/meldung/42578 [24. Aug 2004].

Kuri, J. (2004), 'Musikindustrie will Online-Plattform Phonoline aufgeben [Update]', heise online http://www.heise.de/newsticker/meldung/51482 [28. Sep 2004].

Land Hessen (1946), 'Verfassung des Landes Hessen (HLV)', http://www.hessenrecht.hessen.de/gvbl/gesetze/10_1Verfassung/10-1-verfass/verfass.htm [30. Sep 2004].

Lau, P. (2002), 'Musik für Erwachsene (2)', *brand eins* **9**, S. 51 ff. http://www.brandeins.de/home/inhalt_detail.asp?id=325&MenuID=130&MagID=11&sid=su2179316192852774 [14. Nov 2004].

Lessig, L. (2004), *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, The Penguin Press, New York, NY. http://www.free-culture.cc/freeculture.pdf [01. Nov 2004].

M/A/R/R/S (1987), 'Pump Up The Volume', 4AD AD70.

Microsoft Deutschland (2004), 'Wie funktioniert DRM?', http://www.microsoft.com/germany/digital-mentality/funktionsweise.mspx [30. Sep 2004].

Netzeitung (2004), 'Fusion von Sony Music und BMG perfekt', *Netzeitung* **6**. http://www.netzeitung.de/wirtschaft/unternehmen/299070.html [15. Sep 2004].

New Order (1983), 'Blue Monday', Factory FAC73.

Oechsler, J. (1998), 'Skript zum Urheberrecht und gewerblichen Rechtsschutz'. http://www.uni-potsdam.de/u/ls_oechsler/lehre/skript/ur.pdf [01. Nov 2004].

Portishead (1994), 'Sour Times', Go Beat GODCD116.

Stallman, R. (2002), 'Can you trust your computer?', *News Forge* . http://www.newsforge.com/business/02/10/21/1449250.shtml?tid=19 [19. Okt 2004].

Stein, S. (2004), 'Drei Jahre Knast für Tauschbörsenbenutzer!', *Computerbild* **20**.

Tóth, G. (2004), 'DRM and privacy – friends or foes?', *INDICARE Monitor* **1**(4). http://indicare.berlecon.de/tiki-read_article.php?articleId=45 [20. Okt 2004].

Wang, X. (2004), 'MPEG-21 Rights Expression Language: Enabling Interoperable Digital Rights Management', http://www.computer.org/multimedia/mpeg.htm [20. Okt 2004].

Wicke, P. (1997), 'Musikindustrie im Überblick – Ein historisch-systematischer Abriß'.

Wilkens, A. (2004), 'Online-Tauschbörsen sind beliebter denn je', heise online http://www.heise.de/newsticker/meldung/49023 [16. Nov 2004].

World Intellectual Property Organization (WIPO) (1996), 'WIPO Performances and Phonograms Treaty, Geneva', http://www.wipo.int/documents/en/diplconf/distrib/95dc.htm [20. Sep 2004].

Das Netlabel als alternativer Ansatz der Musikdistribution

SEBASTIAN REDENZ

Schon lange vor der Entwicklung von Peer-to-Peer-Netzwerken und Bezahldownloads gab es virtuelle Musiklabel, die legal digitale Musikdateien anboten. Unabhängig von rückgängigen Verkaufszahlen in der Musikbranche, arbeiten Netlabels konsequent an einem alternativen Vertriebsmodell für Musik. Wie dies unter rechtlich legalen Rahmenbedingungen bereits seit Jahren geschieht, soll der vorliegende Beitrag verdeutlichen, der die Motivation und Systematik des vom Autor geleiteten Netlabels *Thinner*[1] beschreibt. Hierbei untersucht der Artikel die historische Entwicklung der Netlabels und benennt deren Ursprünge. Unter Berücksichtigung kausaler Zusammenhänge, die zur aktuellen wirtschaftlichen Depression innerhalb der *Independent Musikindustrie* für elektronische Musik führten, werden Gemeinsamkeiten und Unterschiede zwischen konventionellen Musiklabels[2] und Netlabels und daraus resultierende Konsequenzen erörtert, während abschließend Chancen und Risiken letzterer genannt werden.

1. Einleitung

Ein Netlabel ist eine unabhängige Online-Plattform, die MP3- oder OGG-Musikdateien[3] zum kostenfreien Download im Internet anbietet. Der Großteil aller Netlabels veröffentlicht elektronische Musik, da die strukturellen Grundlagen der Netlabels erst durch die Entwicklung der Heimcomputertechnologie mit der parallelen Kommerzialisierung des Internets ermöglicht wurden. Deswegen ist in diesem Artikel das Hauptaugenmerk auf diese Musikrichtung gerichtet und es werden deren Besonderheiten untersucht. Das 2001 gegründete *Thinner* Netlabel vertreibt Musikdateien, die verschiedenen Genren der elektronischen Musik zugeordnet werden. Der Tonträgermarkt für elektronische Musik zeichnet seine Vielfältigkeit durch das Vorhandensein vieler Subgenres aus. Im Vergleich zu populären Genres wie Rock- und Popmusik sind die Umsätze im Bereich der elektronischen Musik jedoch als gering einzustufen.

3 OGG Vorbis ist ein Kompressionsformat unter Open-Source-Lizenz. Es ähnelt dem MP3-Format, wobei jedoch keine Patente darauf gehalten werden.

Netlabels stellen heutzutage einen der wenigen Wachstumsmärkte für elektronische Musik dar und erfreuen sich zunehmender Akzeptanz. Vermehrt gelingt es Netlabels, mit Veröffentlichungen etablierter Künstler[4] auch außerhalb des Internets Aufmerksamkeit zu erreichen. Fernziel des Netlabels ist es, dass eigene Veröffentlichungen außerhalb des Internets dieselbe Aufmerksamkeit und Relevanz wie Veröffentlichungen konventioneller Label erfahren. Auf dem Weg dorthin sind die Lizenzmodelle von Creative Commons[5] von entscheidender Bedeutung.

2. Geschichte der Netlabels – Das Tracker-Phänomen

Die Ursprünge der heutigen Netlabelszene reichen bis in das Jahr 1987 zurück, als der seinerzeit bekannte Computermusiker Karsten Obarski ein kleines Programm für den Commodore Amiga namens *Ultimate Soundtracker*[6] entwickelte, mit welchem man in der Lage war, primitive Songmodule (engl.: „Mods“[7]) zu programmieren. Diese Art des Musikmachens erfolgte intuitiv, auch dadurch begünstigt, dass Notenlesen keine elementare Voraussetzung für die Bedienung des Programms war. Die klangliche Qualität war jedoch eingeschränkt, daher wurden Tracker-Programme dann verwendet, wenn Klang und Dynamik der geplanten Produktion eine sekundäre Rolle spielten. Das Ansehen der Tracker-Programme wurde dadurch begründet, dass man nach wenigen Stunden Einarbeitung durchaus in der Lage sein konnte, eingängige und kurzweilige Songmodule zu programmieren.

1990 erschien das Programm *Pro Tracker* für den Amiga, das als Ausgabe ebenfalls Mods generierte. Pro Tracker erschien als *Freeware* und wurde das am weitesten verbreitete Tracker-Programm, um welches sich eine relativ große Fangemeinde im Internet aufbaute. Der Hauptgrund für die Popularität der Tracker-Programme war der Open-Source-Charakter der Mods – zwar gab es Abspielsoftware, jedoch war es auch möglich, die Mods direkt in den Pro-Tracker zu laden. Dort erhielt man dann direkten Einblick in die Song- und Patternstruktur[8] und konnte das Musikstück praktisch wie vom Notenblatt mitlesen, aber auch bearbeiten.

Um die Mods entstand eine Subkultur, in der es Mitte der 90er Jahre zu vermehrten Gründungen sog. „Module Groups“[9] kam, die regelmäßig Mods veröffentlichten.

4 Ein Künstler gilt in diesem Kontext als „etabliert“, sofern dieser eine umfangreiche Discographie, bestehend aus zahlreichen Veröffentlichungen konventioneller Tonträger, nachweisen kann.

5 Das Lizenzmodell Creative Commons wird in den Abschnitten 8. und 9. behandelt.

6 Der *Ultimate Soundtracker* war das erste Tracker-Programm. Ein „Tracker-Programm“ (abgeleitet von Track; techn. Audruck für „Tonspur“) war in der Lage, Samples auf einer schrittweisen Zeitleiste über mehrere einstimmige Kanäle anzuordnen. Die Eingabe der Tonhöhe, Lautstärke und Effektbefehle erfolgt dabei alphanumerisch in Tabellen, teilweise sogar in hexadezimaler Schreibweise. Ein fertiges Musikstück besteht aus mehreren mehrstimmigen Abschnitten, sogenannten „Patterns“, die mittels einer Liste hintereinander gehängt werden (Wikipedia 2005).

7 Ein „Modul“ oder „Mod“ ist eine Datei, in welcher Notation, Samples und sonstige Daten eines Stückes zusammengefasst sind. Die Arbeitsumgebung zur Erstellung von Mods bildet das Tracker-Programm.

8 Repetitives Muster nach dem Etwas zeitlich abläuft. Im Zusammenhang mit tanzbarer elektronischer Musik spricht man auch von „Loop“.

9 Module Groups waren die Vorläufer heutiger Netlabels und existierten Mitte bis Ende der 90er Jahre. Bekannteste Vertreter waren *Radical Rhythms* (http://rr.rockz.org), *Kosmic Free Music Foundation*

Module Groups bestanden i. d. R. aus einem Stamm von fünf bis zehn aktiven Musikern. Im Allgemeinen besaßen die Mitglieder solcher Module Groups Fähigkeiten im Programmieren und im Umgang mit Tracker-Programmen. Module Groups agierten in einer kohärenten Szene, zumeist ohne Kontakt zum realen Musikgeschäft und vertrieben ihre Produktionen über spezielle *Newsgroups* und *BBS-Mailboxen*. So fungierten sie quasi als Vorhut der heutigen Netlabels (Röttgers 2003, S. 129 f.).

Tracker-Programme stellten Nischenprogramme für computerbegeisterte Menschen dar, die eine gewisse Faszination für den synthetischen Klang der damaligen Heimcomputer vereinte. Für professionelle Musiker waren Tracker-Programme jedoch meist uninteressant, da die klanglichen Möglichkeiten sehr limitiert waren. Wollte man professionell Musik machen, die sich kommerziell verwerten ließ, galt es, ein Studio zusammenzustellen.

3. MP3-Format und Wandel in der Studiotechnologie

Der Chronologie nach entwickelte das Fraunhofer Institut ab dem Jahre 1987 das MP3-Format. Durch komplexe Algorithmen wurde es erstmals möglich, binäre Musikdaten in CD Qualität um das 12fache zu komprimieren, um eine verlustbehaftete, jedoch akzeptable Kompression zu erhalten. Der sonische Verlust betrifft dabei Frequenzen, die vom menschlichen Ohr nicht wahrgenommen werden (Fraunhofer Institut 2005).

Basierend auf dem Fraunhofer Algorithmus tauchte ab 1997 Enkodierungssoftware im Internet auf, die von Dritten programmiert und angeboten wurde. Mit einem solchen MP3-Encoder war der Anwender in der Lage, eine akkurate binäre Kopie einer CD binnen weniger Minuten anzufertigen. In akzeptabler Enkodierungsstufe sind MP3-Dateien meist nur wenige Megabyte groß. MP3-Dateien wurde in den neu entstehenden Tauschbörsen wie Napster zum begehrtesten Objekt. Durch die Popularität Napsters wurde früh klar, dass sich das MP3-Format durchsetzen und zum Standard für digitale Musikdateien etablieren wird.

Parallel zum Phänomen MP3 nahm die technische Entwicklung der Heimcomputer rasant zu, während die Preise für selbige Komponenten stetig fielen. Der Computer als Heimstudio wurde für professionelle Musiker nun eine ernsthafte Alternative, und viele zögerten auch nicht, ihre analogen Klangerzeuger zu verkaufen und die Grundlage ihrer Produktionsmöglichkeiten komplett auf moderne Software umzustellen.

Es bildete sich ein neuer Markt für Produktionssoftware, wobei Programme wie *Reaktor* oder *Reason*[10] in der Lage sind, Schaltkreise klassischer synthetischer Klangerzeuger mit einem hohen Realitätsgrad zu emulieren. Darüber hinaus bieten sie Konstruktionsmöglichkeiten für eigene virtuelle Klangmodule an. Ein komplettes Studio, ausgestattet mit allerlei phyischen Klangerzeugern, konnte fortan vollständig ersetzt werden.

(KFMF) und die heute noch existierende Group *Monotonik* (http://www.monotonik.com).

10 Siehe hierzu http://www.nativeinstruments.com (Reaktor) und http://www.propellerheads.com (Reason)

Ließ sich die Dateigröße eines Mods noch anhand der Anzahl und Dateigröße der enthaltenen Samples bestimmen, verhält es sich beim MP3 anders: Die Größe richtet sich nach Länge und Kompressionsqualität des jeweiligen Stücks. Ein 6-minütiges Stück kann bei guter Qualität schnell 8–10 Megabyte umfassen. Diese Tatsache schreckte viele Anhänger der Mods zunächst ab, mussten sie für den Download eines einzigen MP3s die gleiche Zeit einräumen wie zum Download mehrerer Dutzend Mods. In Zeiten, als die Internetverbindung der meisten privaten Haushalte durch das analoge Modem hergestellt wurde, konnte der MP3-Download somit schnell zu einer kostenintensiven Angelegenheit werden.

Dennoch setzte sich das MP3-Format in kurzer Zeit durch und wurde zum Standard für digitale Musikdateien. Nur noch wenige Musiker konnten sich für Tracker-Programme begeistern, was das Ende der allermeisten Module Groups bedeutete. Nur wenige Module Groups passten sich den neuen Gegebenheiten an und wechselten das Audioformat.

4. Aufstieg der Netlabels

Mitte der 90er Jahre gab es Module Groups, die als loser Verbund zwischen Musikern und Programmierern organisiert waren. Das Ziel solcher Organisationen war es, Hobbymusiker zu fördern, indem man ihnen eine Plattform bot, auf der sie frei von kommerziellen Zwängen ihre Produktionen veröffentlichen konnten.

Das MP3-Format etablierte sich 1998 als einheitliches Format für Computermusik. Für Betreiber von Module Groups boten sich somit neue Möglichkeiten, da sie nun mit Musikern arbeiten konnten, die nicht der Tracker-Szene entstammten. Zudem haben MP3-Dateien eine bessere klangliche Qualität als Mod-Dateien. Aufgrund dieser Begebenheiten entschieden sich einige Module Groups, fortan im MP3-Format zu veröffentlichen und auf Mods zu verzichten. Mit dem Wechsel des Formats änderte sich auch die Bezeichnung der Organisationen – aus Module Groups wurden Netlabels.

Ziel eines Netlabels ist es, Künstler systematisch in ihrer musikalischen Entwicklung zu fördern und deren Veröffentlichungen zu bewerben. In der Regel stellt das Netlabel ein Künstlerrepertoire zusammen, welches dem Label einen übergeordneten Musikstil verleiht. Ein Netlabel veröffentlicht in Intervallen und selektiert seine Veröffentlichungen im Vorfeld anhand qualitativer Kriterien. Netlabels sind nicht kommerziell, werden ehrenamtlich betrieben und funktionieren primär als Künstlerkollektiv mit hierarchischen Strukturen (Röttgers 2003, S. 140 f.).

Im Gegensatz dazu stehen Onlineportale, wie z. B. *MP3.com*[11] oder *Besonic.com*[12]. Jene Portale verstehen sich als Katalog für sämtliche Musikrichtungen und bieten MP3s

11 *MP3.com* war Ende der 90er Jahre das bekannteste börsendotierte MP3-Portal und wurde schließlich nach mehreren Gerichtsprozessen im Jahre 2001 für 372 Millionen US-Dollar an *Vivendi Universal* verkauft. 2002 verkaufte Universal das mittlerweile inaktive MP3.com Portal weiter an *CNet*. Jedoch wurde MP3.com trotz zahlreicher Ankündigungen bis heute nicht reaktiviert.

12 *Besonic.com* ist ein deutsches MP3-Portal, welches 2003 Insolvenz anmelden musste und sich mittlerweile im Besitz von *Terratec* befindet.

von Künstlern an, die sich kostenlos registrieren und ihre Werke auf die Portalseite laden. Ihren Profit erwirtschaften MP3-Portale i. d. R. durch Werbeeinnahmen und durch den Verkauf von CDs der Künstler. Die Künstler werden am Erlös dieser CDs beteiligt. Das Interesse einer Portalseite am Künstler ist stets ein rein kommerzielles.

5. Konventionelles Label vs. Netlabel

Da elektronische Musik in den Nischenbereich fällt, ist der Markt hart umkämpft. Die Auflagengröße bei Independent Labels für elektronische Musik pendelt heutzutage zumeist zwischen 500 und 1 500 Einheiten – größere Absatzmengen bilden die Ausnahme. Die geringen Absatzzahlen führen folglich zu kleinen Gewinnmargen.

Für die meisten Künstler hat eine Veröffentlichung auf Vinyl, neben der Genugtuung, physisch verewigt zu werden, einen gewissen Referenz-Charakter. Beim Versand eines Demos für zukünftige Tonträgerveröffentlichungen kann somit implizit auf vorherige Veröffentlichungen verwiesen werden. Zusätzlich gilt es als gute Werbung für Konzerte oder DJ-Auftritte.

Zwischen 1999 und 2001 stieg die Zahl der Gründungen von Plattenfirmen für elektronische Musik enorm an. Auf dem ohnehin schon unelastischen Markt herrschte ein Überangebot an Veröffentlichungen, sodass besonders kleinere Labels Absatzschwierigkeiten hatten, da die Nachfrage primär von etablierten Labels bedient wurde. Zudem wurde der Verbreitungsgrad von MP3 in Zeiten der Flatrate unterschätzt, da immer mehr Menschen Kenntnis darüber erlangten, wie und wo man im Internet die gewünschten MP3s findet.

So wurden immer mehr beschreibbare Medien wie die CD-R abgesetzt, während der Verkauf konventioneller CDs abnahm. In den Augen vieler Konsumenten stimmte das Preis/Leistungsverhältnis nicht mehr, da es nur wenige Mausklicks benötigte, einige Lieder oder gleich das ganze Album im Internet zu finden, selbiges zu kopieren und auf eine abspielbare CD zu übertragen. Erfolgreiche Konzepte, wie den sinkenden Absatzzahlen gegenzusteuern ist, konnten nicht in die Tat umgesetzt werden, sodass in der Konsequenz größere Vertriebe wie *Fine Audio*, *Prime* und *E. F. A. Medien* Insolvenz anmelden mussten (Balzer 2004). Dies bedeutete gleichzeitig das Aus für viele Independent Labels, die mit den betroffenen Vertrieben im Vertragsverhältnis standen. Verunsichert von den Ereignissen, reagierten viele Independent Labels mit abnehmender Veröffentlichungsfrequenz. Leidtragende sind hierbei die Künstler, da aufgrund des schwierigen Marktes vertragsbasierte Künstler–Label-Beziehungen oftmals aufgelöst werden mussten.

Für den Künstler stellt sich, aktueller denn je, die Frage, ob es nicht eine sinnvolle Alternative ist, die Musik aus eigenen Stücken im Internet zu veröffentlichen, da ein Netlabel im Vergleich zu einem konventionellen Label viele Vorteile bietet (siehe hierzu Tabelle 1).

	Netlabel	Physisches Label
Form	Binär; in Form von MP3/OGG Dateien	Physisch; CD/LP
Grafik	Interaktives/Moduliertes Artwork	Statisches Cover
Auflage	Unlimitiert; wahlweise temporär	Limitiert; in seltenen Fällen erneute Auflage
Investitionen	Sehr geringe finanziellen Hürden	Sehr hohe Aufwendungen notwendig
Distribution	Entfällt	Spezielle Läden (in größeren Städten); Mailorder
Effizienz	Selbstkontrolle über Downloads	Verlass auf Aussagen Dritter (Vertrieb)
Radius	Global	Hauptsächlich Industrieländer
Preis	Onlinezugang, Computer	8–18 Euro je Tonträger; zusätzl. Investitionen in Abspielgerät
Vergütung	Micropayment; Selbstvertrieb von CD-Rs	Prozentsätze; öfters auch Abzug der Promotions-Aufwendungen

Tabelle 1: Konventionelles Label vs. Netlabel

6. Distribution und Radius

Dem Jahresbericht 2003 der *International Federation of the Phonographic Industry (IFPI)* lässt sich anhand der Aufteilung der verschiedenen Altersgruppen von Musikkäufern entnehmen, dass elektronische Musik vor allem von den 10–19-jährigen und 20–29-jährigen nachgefragt wird. Laut IFPI haben diese Altersgruppen weitgehenden Zugang zu Computern, CD-Brennern und Internet und substituieren nachlassende Käufe durch Kopien. Dies zeigt, dass Tonträger mit einem variablen Preis zwischen 8–18 € in einem dynamischen Markt, in dem Veröffentlichungen meist von kurzer Aktualität sind, schnell zu einem Luxusgut werden (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 36).

Zwar ist der Markt mit einer Million abgesetzter Vinylschallplatten im Jahr 2003 im Vergleich zu den Vorjahren konstant geblieben, jedoch sind die Auflagenhöhen je Veröffentlichung der einzelnen Labels aufgrund der hohen Dichte an Marktteilnehmern gering (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 25). Meist bleibt es bei Erstauflagen.

Für den Künstler stellt sich die Frage, ob es für ihn sinnvoller ist, auf einem Label zu veröffentlichen, welches lediglich 750–1000 Kopien Tonträger in Umlauf bringt, oder ob er auf einem Netlabel veröffentlicht, wo die Werke stets präsent und verfügbar sind.

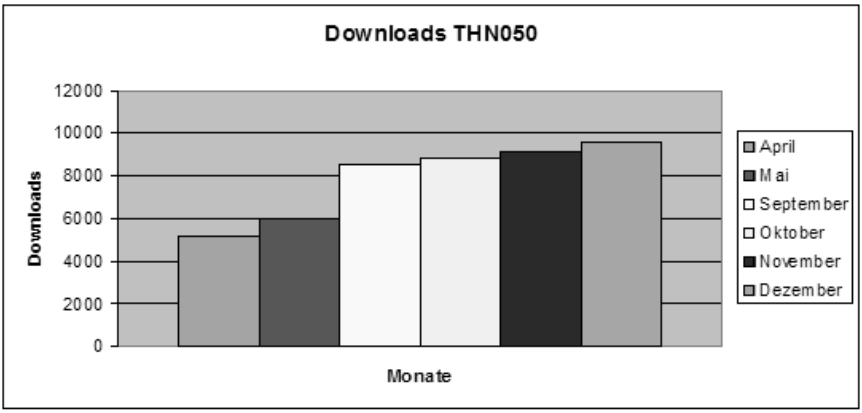


Abbildung 1: Downloads der Veröffentlichung „I Like To Listen!" (THN050 2004)

Abbildung 1 zeigt das arithmetische Mittel der Downloads je Song der Veröffentlichung „I Like To Listen!" (THN050 2004) auf Thinner bis zum Dezember 2004 an. Es werden demnach kontinuierlich neue Hörer angelockt, und die Anzahl der Downloads übertrifft bei weitem die Auflage eines konventionellen Mediums. Durch die Distribution im Internet können auch Hörer aus Ländern erreicht werden, in denen elektronische Musik unterrepräsentiert und nur schwer zu bekommen ist. Dies betrifft vor allem ökonomisch schwächere Länder.

Eine solche Präsenz wird durch das Auslagern der MP3-Archive auf öffentliche FTP-Seiten wie *Scene.org* oder *Archive.org* ermöglicht (Internet Archive 2005, The International Scene Organization 2005). Diese Archive bieten Speicherplatz für verschiedene multimediale Künste und sind nicht-kommerziell. Rechenzentren, die von Universitäten betrieben werden oder Stiftungen, wie *Hewlett.org*, stellen die notwendige Infrastruktur oder garantieren mit großzügigen Spenden für den Fortbestand solcher FTP-Archive. Netlabel-MP3s tauchen aber auch oft in Tauschbörsen wie *Soulseek* oder *BitTorrent* auf, und dank der Creative-Commons-Lizenz[13] geschieht dies im Einklang mit Künstler und Label.

Jedoch darf an dieser Stelle nicht unerwähnt bleiben, dass die Netlabels von öffentlichen FTP-Archiven abhängig sind. Sollten eine oder beide der oben genannten Seiten die Arbeit und Unterstützung einstellen, dürfte dies das Ende derjenigen Netlabels bedeuten, die mit ihrem hohen Traffic-Aufkommen an der Spitze der Subkultur stehen. Ein Netlabel wie Thinner hat inkl. dem Sublabel Autoplate ein geschätztes Traffic-Aufkommen von monatlich einem halben Terabyte. Müssten die Serverkosten hierfür aus eigenen Mitteln gedeckt werden, dürfte dies nur schwer zu finanzieren sein.

13 Das Lizenzmodell Creative Commons wird in den Abschnitten 8. und 9. behandelt.

7. Micropayment, Merchandise und Konzerte als Einnahmequellen

Als Musiker möchte man vergütet werden, sollte ein Label die Werke des Künstlers kommerziell auswerten wollen. Dies geschieht, indem es industriell gefertigte Tonträger in Umlauf bringt. Bei den geringen Auflagen im Independent-Sektor können die Erlöse aus dem Tonträgerverkauf meist jedoch nur die unmittelbaren Kosten decken.[14] Die Gewinne für den Künstler fallen daher meist gering aus und übersteigen bei einer Auflage von circa 1 000 abgesetzten Einheiten selten 500 Euro.

Während der Verkauf die Geschäftsgrundlage eines jeden physischen Labels darstellt, findet ein Tonträgerverkauf bei einem Netlabel nur in seltenen Fällen statt. Bei *Thinner/Autoplate* wird jedem Künstler selbst überlassen, ob er in Eigenregie eine limitierte, optional ergänzte, konventionelle Auflage anbieten will. Die Erfahrung zeigt, dass ein Künstler mit einer Auflage von 100 CD-Rs, die in Eigenregie produziert und abgesetzt werden, einen ungleich höheren Profit erwirtschaften kann, als durch eine Veröffentlichung auf einem Label mit 1 000er Vinyl Auflage. Dies wird möglich, da das Netlabel auf eine Gewinnbeteiligung verzichtet. Zusätzliche Einnahmen bieten sich dem Künstler durch Geldspenden an.[15]

Für das Netlabel, welches u. U. ein gewisses monatliches Budget zur Bewerbung seiner Veröffentlichungen veranschlagt, bieten sich Einnahmequellen durch den Verkauf von T-Shirts oder Sammlungen. Internet-Nutzer ohne Flatrate zeigen wenig Bereitschaft, das komplette Archiv eines Netlabels herunterzuladen, sollte es einen gewissen Umfang überschreiten. Dem Besucher bietet sich eine Zeit- und evtl. auch eine Kostenersparnis, wenn er die Option hat, den kompletten Katalog eines Labels auf einem Datenträger zu erwerben. Von zusätzlicher Attraktivität für ein solches Angebot könnten exklusive Zugaben in Form von unveröffentlichten Stücken, Videos und gesammelten Livekonzerten sein. Zusätzlich kann der Datenträger auf eine geringe Anzahl limitiert sein.

Primäre Einnahmequellen für Künstler stellen im Allgemeinen Konzerte dar. Viele Netlabel-Betreiber haben jedoch keine Absichten, Konzerte für ihre Künstler zu vermitteln und möchten sich auf das Veröffentlichen von Musik beschränken. Ein weiterer Punkt ist, dass viele Clubbesitzer der jungen Netlabel-Kultur gegenüber eine reservierte Haltung einnehmen. Dies liegt darin begründet, dass die Präsenz in den Medien noch relativ gering und unregelmäßig ist und dadurch ein größeres finanzielles Risiko für die Clubbetreiber impliziert wird.

Etablierte Labels, die regelmäßig vierstellige Absätze ihrer Tonträger erreichen, bewerben ihr Label und ihre Produkte, indem sie den jeweiligen Künstlern Konzerte oder ausgedehntere Tourneen vermitteln. Sie weisen ein geringes Risiko auf, indem die

14 Der größte Teil des Kostenbudgets veranschlagt die Herstellung der Tonträger. Zusätzliche Kosten entstehen durch Bewerbung der Produkte, beispielsweise durch Anzeigen in Magazinen oder den Versand von Tonträgern, die der Promotion dienen. Weitere Kosten können für das Design und Mastering der Tonträger anfallen.

15 Firmen wie *Ebay* (PayPal) und *Ikobo* ermöglichen es dem Konsumenten, eine direkte Spende (engl. Fachausdruck: „Micropayment") entweder an das Label oder an den Künstler zu leisten. Thinner war das erste Netlabel, das diese Praxis eingeführt hat.

Gagen fix bemessen werden und sich nach der Größe der Veranstaltungsorte richten. In der tanzbaren elektronischen Musik sind dies vor allem Nachtclubs in größeren Städten, in denen Liveauftritte stattfinden.

8. Zur Notwendigkeit eines Lizenzmodells bei Veröffentlichungen im Netz

In der Vergangenheit traten vereinzelt Fälle auf, wo Netlabel Veröffentlichungen von konventionellen Labels vereinnahmt wurden. Teilweise wurden sogar Künstler mit Scheinverträgen gelockt und die gesamten Musikstücke einiger einzelner Veröffentlichungen des Netlabels beansprucht. Zudem wurden MP3s ohne vorherige Abklärung unter falschem Namen weiterverkauft. Um besser gegen solche Eingriffe geschützt zu sein, sollte es im Sinne sowohl des Künstlers als auch des Netlabels sein, die Veröffentlichungen unter eine Lizenz zu stellen. Zusätzlich sollte man Leitlinien an die Künstler vorgeben, die die Umstände einer möglichen Lizenzierung an Dritte definieren.

Das klassische Urheberrecht wirkt sich für ein Netlabel vor allem restriktiv aus, denn dieses sieht vor, dass Dritte keine Legitimation haben, Werke eines Künstlers von einer Website herunterzuladen und weiterzugeben – dieses Recht obliegt nur dem Autor selbst. Es gibt aber auch Künstler, die der Verbreitung ihrer Werke zustimmen würden, jedoch gab es für diesen Fall keinen rechtlichen Rahmen. Lawrence Lessig, seines Zeichens Professor an der Universität Stanford, beschäftigt sich seit Jahren mit der Umsetzung einer liberaleren Handhabung des Urheberrechts. Seinen Aussagen nach sei ein *„Copyright-Krieg"* im Gang, *„der Fortschritt, Innovation, Lernen und Wachstum gefährdet"* (zitiert in Krempl 2004).

Zur Lösung des Problems tat sich Lessig mit kompetenten Wissenschaftlern verschiedener Teilbereiche des öffentlichen bzw. Medienrechts zusammen, um Abhilfe zu entwickeln. Ziel war es, verschiedene Urheberrechtsmodule zu entwickeln und zur Verfügung zu stellen, die dem Urheber eines Werkes ermöglichen, den öffentlichen Umgang mit seinem Werk selbst zu bestimmen. Das Resultat war die Initiative „Creative Commons".

Creative-Commons-Lizenzen erlauben Reproduktion unter gewissen Bedingungen, die für jedes Werk einzeln anpassbar sind. Netlabels entscheiden sich in den meisten Fällen für folgendes Modell:

- Namensnennung: Der Verbraucher muss bei öffentlicher Aufführung des Stücks den Produzenten des Werkes nennen.
- Keine kommerzielle Nutzung: Die Inhalte dürfen nicht für kommerzielle Zwecke verwendet werden.
- Keine Bearbeitung: Der Inhalt darf nicht bearbeitet oder in anderer Weise verändert werden.

Vor allem unbekannte Künstler setzen Creative-Commons-Lizenzen ein, wenn sie auf ihren Internetseiten eigene Stücke bewerben (Jaensch 2004).

9. Creative Commons vs. GEMA

Viele Künstler sind Mitglied einer Verwertungsgesellschaft; in Deutschland ist dies die *Gesellschaft für musikalische Aufführungs- und mechanische Vervielfältigungsrechte* (GEMA), welche nach dem Ausschließlichkeitsprinzip arbeitet. Die GEMA gestattet ihren Mitgliedern, sich aus bestimmten Sektoren der Vergütung, bspw. der Online-Verwertung, „befreien" zu lassen. So ist es für den Künstler möglich, unter Bezugnahme auf eine Creative-Commons-Lizenz auf einem Netlabel zu veröffentlichen.

Jedoch muss der Künstler hier vorsichtig sein. Die Creative-Commons-lizenzierten Werke dürften während eines Konzertes nicht im Repertoire enthalten sein. Denn sollten die Aufführungs- und Senderechte der Werke des Künstlers weiterhin der GEMA aufgetragen sein, würde jedes öffentlich aufgeführte Werk in deren Verwertungsbereich fallen; einzelne Ausnahmen werden bei der GEMA nicht gemacht. Somit wäre das ein Verstoß des Künstlers gegen die GEMA, der in Konsequenz zu einem Ausschluss aus der Gesellschaft führen könnte.

Ein anderer Knackpunkt ist die *Creative Commons Music Sharing License*, die es dem Künstler ermöglicht, über Werke Verträge zu schließen, die vorher bereits unter der Bedingung der „nicht-kommerziellen Nutzung" lizenziert wurden. Hat man jedoch als GEMA-Mitglied diesbezüglich vorher schon Ausschüttungen erhalten, wird es problematisch, denn die GEMA besteht auf Einhaltung des Ausschließlichkeitsprinzips. Sie duldet keine weiteren Lizenzen in Bereichen, in denen sie im Vorfeld bereits authorisiert wurde, die Rechte des Mitglieds auszuüben (Herrmann und Kösch 2004).

Im Juni 2004 wurde eine Arbeitsgruppe gebildet, die es sich zum Ziel gemacht hat, die Creative-Commons-Lizenzmodule vollständig konsistent in Bezug auf deutsches Recht auszuformulieren.[16] Dabei wird der Dialog mit der GEMA gesucht, um für deren Mitglieder gemeinsam nach kompetenten Lösungen zu suchen, damit diese im Zusammenspiel mit beiden Organisationen in Zukunft den Überblick behalten. Hierbei gilt es, die Resultate abzuwarten.

10. Zusammenfassung und Empfehlungen

Die Netlabel-Kultur steht in Bezug auf öffentliche Aufmerksamkeit erst am Anfang. Obwohl Netlabels bereits seit Jahren existieren, konnte der rechtliche Rahmen mit den Creative-Commons-Lizenzen erst vor relativ kurzer Zeit gebildet werden. Jedoch werden heute schon Modell und Funktionsweise des Netlabels von sogenannten „Digital Labels"[17] kopiert. Diese bieten MP3-Dateien kostenpflichtig an und vertreiben ihren Katalog in der Regel über Online-MP3-Shops, wie z. B. *Nufonix.com* oder *Beatport.com*. Zwar genießt die Vinylschallplatte als Medium innerhalb der Independent–Musikindustrie nach wie vor einen großen ideellen Wert, jedoch werden angesichts wirtschaftlicher Zwänge viele Labels in Zukunft auf digitale Formate setzen. Dieser Wandel wird jedoch nicht zwangsläufig an elektronische Musik gebunden sein. Auch

16 Siehe http://icommons.org.

17 Ein solches Digital-Label findet man zum Beispiel unter http://www.jeffbennett.info/kungfudub.

außerhalb davon dürften Idee und Funktionsweise des Netlabels auf Interesse stoßen und umgesetzt werden.

Jedes Netlabel könnte mehr individuelle Aufmerksamkeit mit einfachen und traditionellen Mitteln erreichen. Leider wagen es bisher nur wenige Netlabels, Printmedien und Radiostationen regelmäßig zu bemustern. Einerseits ist dies verständlich, da die Musik der Netlabels virtuell angeboten und somit nicht auf einem Tonträger veröffentlicht wird, andererseits nimmt die Mehrheit der musikinteressierten Menschen Musik außerhalb des Internets wahr, indem sie Radio hören und Magazine lesen. Die Erfahrungen mit *Thinner* zeigen, dass durch eine regelmäßige Bemusterung die Besucherzahlen innerhalb weniger Monate verdoppelt werden können.

Möchte man eine hohe monatliche Besucherzahl realisieren, müssen zwangsläufig Strukturen außerhalb des Internets genutzt werden. Da die meisten Journalisten das Musikgeschehen im Internet ignorieren, sollten die Veröffentlichungen des Netlabels die Redaktionen der Musikzeitschriften und Radiosender auf dem Postweg erreichen, um besprochen und öffentlich aufgeführt zu werden.

Als Folge mangelnder Öffentlichkeitsarbeit erscheint vielen Clubbesitzern das Wagnis einer Netlabel-Veranstaltung finanziell zu risikoreich. Somit fällt es vielen Netlabels schwer, erfolgreich Konzerte oder Tourneen zu veranstalten. Viele Netlabels sehen es oft aber auch gar nicht vor, Veranstaltungen zu betreiben. Dies ist verständlich, da die Organisation einer Veranstaltung meist einen großen Zeitaufwand erfordert, den viele Netlabel-Macher nicht aufbringen können, wenn das Netlabel ehrenamtlich betreut wird. Von daher bietet es sich an, dass Netlabels bei möglichen Veranstaltungen kooperieren und sich gegenseitig unterstützen.

Erste Schritte in diese Richtung wurden 2004 mit der Netlabelnacht im Rahmen der Veranstaltung *Evoke* getan, welche von *Tokyo Dawn Records* organisiert wurde und eine Schaubühne für mehrere Netlabels war. Effektiver wäre es jedoch, im Rahmen einer solchen Veranstaltung ausschließlich genreverwandte Netlabels auftreten zu lassen. In diesem Sinne sollte versucht werden, in das Programm größerer Festivals zu gelangen, wie bspw. *Transmediale* (Deutschland), *Sonar* (Spanien) oder *Mutek* (Kanada). Auf dortigen Veranstaltungen könnte sich effektiv an ein breiteres Publikum gewandt werden, wenn man solche Showcases zusätzlich angemessen moderieren würde.

Zu einer verstärkten Wahrnehmung gelangt ein Netlabel aber auch, indem es sich durch kontinuierliche Einhaltung gewisser Konventionen auszeichnet. Dies beinhaltet neben der Einhaltung von hohen MP3-Bitraten, Mastering und einer logischen Struktur bei den Dateinamen und *ID3 Tags* auch die Spezialisierung auf einige wenige musikalische Genres. Viele Netlabels versäumen es an dieser Stelle, sich durch eine gewissenhafte Veröffentlichungspolitik zu profilieren und überfordern die Hörer, indem sie allzu große Sprünge zwischen den Genres wagen. Außerdem sollte es vermieden werden, Stücke zu veröffentlichen, die klanglich nicht professionell überarbeitet wurden. Viele Hörer reagieren darauf mit Befremden und stellen anhand solcher Verfehlungen die generelle Qualität der Netlabels in Frage.

Diesbezüglich ist ein Portal notwendig, welches konsequent und sorgsam Netlabel-Releases nach musikalischer Qualität, basierend auf Einhaltung oben genannter

Kriterien, filtert und für eine breitere Öffentlichkeit durch einfaches und funktionales Design gedacht ist.

Solche Möglichkeiten ließen sich hervorragend dafür nutzen, interessierten Menschen einen qualitativen Einstieg in die digitale Kultur der Netlabels mit regelmäßigen Artikeln und Besprechungen zu ermöglichen. Auf einer solchen Ebene könnte man auch Kooperationen mit Softwarefirmen eingehen. Schließlich sind Netlabels mit den MP3-Dateien für deren DJ-Software-Lösungen[18] von großer Attraktivität.

Da die Creative-Commons-Lizenz bei den Netlabels bekanntermaßen weite Verbreitung findet, sollte auch versucht werden, stärker von Creative Commons miteinbezogen zu werden, wenn es um Panels, Podiumsdiskussionen oder ähnliche Veranstaltungen geht. Schließlich werden durch die Arbeit der Netlabels zehntausende musikinteressierte Menschen mit den Creative-Commons-Lizenzen vertraut gemacht und somit wird öffentlichkeitswirksame Arbeit geleistet.

Literaturverzeichnis

Balzer, J. (2004), 'Schallplattenvertrieb EFA Medien insolvent', *Berliner Zeitung* . Berliner Zeitung vom 03.03.2004.

Bundesverband der Phonographischen Wirtschaft e. V. (2004), 'Jahreswirtschaftsbericht 2003'.

Fraunhofer Institut (2005), 'Audio & Multimedia MPEG Audio Layer-3', http://www.iis.fraunhofer.de/amm/techinf/layer3/ [15. Jan 2005].

Herrmann, T. und Kösch, S. (2004), 'GEMA vs. IFPI – Musik soll sich wieder lohnen', *De:Bug* **83**, S. 28.

Internet Archive (2005), 'About the Internet Archive', http://www.archive.org/about/about.php [15. Jan 2005].

Jaensch, A. (2004), 'Urheberrecht Refreshed – Creative Commons kommt nach Deutschland', *De:Bug* **83**, S. 25.

Krempl, S. (2004), 'Neues Copyright für Kreative – Urheberrecht: Lizenzmodell soll Künstlern mehr Selbstvermarktungschancen öffnen', VDI nachrichten http://www.vdi-nachrichten.com/vdi_nachrichten/aktuelle_ausgabe/akt_ausg_detail.asp?source=rubrik&cat=1&id=17068 [15. Jan 2005].

Röttgers, J. (2003), *Mix, Burn & R. I. P. – Das Ende der Musikindustrie*, Verlag Heinz Heise, Hannover. http://files.lowpass.de/Janko_Roettgers_Mix_Burn_RIP.pdf.

THN050 (2004), 'Various Artists – I Like To Listen!', Thinner Netlabel http://www.thinnerism.com/releases.php?r=thn050 [15. Jan 2005].

The International Scene Organization (2005), 'about scene.org', http://www.scene.org/info.php [15. Jan 2005].

Wikipedia (2005), 'Tracker (Musik)', http://de.wikipedia.org/wiki/Tracker_%28Musik%29 [15. Jan 2005].

18 Wie z. B. die MP3 DJ-Software *Traktor*, siehe http://www.native-instruments.com/index.php?traktor_us

Das Wissen der Welt – Die Wikipedia

PATRICK DANOWSKI UND JAKOB VOSS

Freie Inhalte sind auf dem Vormarsch. Kombiniert mit kollaborativem Arbeiten entstehen außerordentliche Projekte, die es in dieser Form bisher noch nicht gegeben hat. Mit an vorderster Front befindet sich das wohl umfangreichste Projekt dieser Sorte – die Internet-Enzyklopädie Wikipedia. Ermöglicht wird sie durch das recht simple „Wiki-Prinzip", nach dem jeder, ohne Anmeldung oder Filterung durch Redakteure, Inhalte erstellen und bearbeiten kann. Tausende Freiwillige arbeiten aktiv an der Gestaltung der Enzyklopädie und setzen dadurch die einzig bestehende Regel des Systems um – den „Neutralen Standpunkt". Die Autoren beschreiben in dem Artikel die Geschichte der Wikipedia und gehen detailliert auf deren Struktur ein. Zudem erklären sie den alltäglichen Ablauf, wie Beiträge geschaffen werden und wie deren Qualität gesichert werden kann. Abschließend geben sie einen Ausblick über die Zukunft der Enzyklopädie sowie weitere spannende und vielversprechende Projekte mit freien Inhalten. Der Beitrag basiert auf einem im Dezember 2004 in der Ausgabe 8/2004 der Fachzeitschrift „Information – Wissenschaft und Praxis"[1] erschienenen Artikel. Er wurde von den Autoren und Erik Möller in einem Wiki unter http://meta.wikimedia.org/wiki/Open-Source-Jahrbuch/Beitrag_2005 überarbeitet und steht unter den Bedingungen der GNU Free Documentation License und der Creative–Commons-Lizenz „Attributions-ShareAlike" zur Verfügung.

1. Die Wikipedia

Die Wikipedia ist ein internationales Projekt mit dem Ziel, eine freie Internet-Enzyklopädie zu schaffen. Alle Artikel der Wikipedia werden in einem Wiki-System verwaltet, das auf Servern der Wikimedia-Stiftung (*Foundation*) gehostet wird. Die Stiftung wird allein mit Hilfe von Spendengeldern und der Arbeit von Tausenden Freiwilligen auf der ganzen Erde betrieben. Ursprünglich im Januar 2001 als englischsprachiges Projekt gegründet, existieren inzwischen Ableger in mehr als 80 Sprachen. Die deutschsprachige Wikipedia ist nach der englischen die umfangreichste. Mit mehr als einer Million

1 http://www.dgd.de/dgi/nfd/

Artikeln ist die gesamte Wikipedia größer als jede andere Enzyklopädie und gehört laut *Alexa.com* inzwischen zu den 150 populärsten Webseiten weltweit.

2. Das Wiki-Konzept

Wikis sind Webseiten, die nicht zwischen Schreib- und Lesezugriff differenzieren: Wer sie lesen kann, der darf auch Seiten bearbeiten und anlegen. Dazu enthält jede Seite eines Wikis einen Link, durch den sich ein Dialog öffnet, in dem sich der Inhalt der Seite bearbeiten lässt. Dies geschieht in einer vereinfachten Syntax, die es auch ohne Kenntnisse von HTML erlaubt, Formatierungen und Links zu anderen Seiten des Wikis oder ins Internet anzulegen. Alle Bearbeitungsschritte werden gespeichert, so dass alle Änderungen mitverfolgt werden können. Anstatt Änderungen und Neueintragungen zunächst von einem Herausgeber oder Experten begutachten zu lassen (Zensur bzw. *Peer Review*) findet die Kontrolle in Form von Kommentaren und weiteren Änderungen im Nachhinein statt.

Das erste Wiki wurde 1995 von Ward Cunningham als Werkzeug zum Wissensmanagement entwickelt und in Anspielung auf das World Wide Web (WWW) und die hawaiianische Bezeichnung „wiki wiki" für „schnell" das WikiWikiWeb genannt. Durch das Prinzip, die Bearbeitung und Verlinkung von einzelnen Seiten so einfach wie möglich zu machen, eignen sich Wikis hervorragend, um in Gruppen semistrukturierte Text- und Wissenssammlungen zu verwalten. Weitere Einzelheiten zum Aufbau eines Wikis siehe Cunningham (2001).

Für die Wikipedia wurde anfänglichst die Software *UseModWiki* von Clifford Adams verwendet und später durch ein von Magnus Manske eigens für die Wikipedia entwickeltes Programm ersetzt. Inzwischen ist die Wikipedia das mit Abstand größte Wiki, und ihre Software MediaWiki wird von einem Team von Entwicklern als freie Software weiterentwickelt.

3. Freie Inhalte

Dass die Wikipedia frei ist bedeutet, dass ihr gesamter Text den Bedingungen der *GNU Free Documentation License* (GFDL)[2] unterliegt. Die wesentlichen Elemente dieser Lizenz sind, dass die Inhalte frei verbreitet und in neuen Werken weiterverwendet werden können. Diese neuen Werke müssen jedoch ebenfalls wieder unter die Lizenz gestellt und mit einer Quellenangabe zu den ursprünglichen Texten versehen werden. Dieses Prinzip, bei dem jedem die Freiheit zum Nutzen, zum Verändern und zum Weiterverbreiten der Inhalte eingeräumt wird, stammt aus der Bewegung für freie Software.

Da die GFDL ursprünglich für technische Dokumentationen und nicht für Wikis entwickelt wurde und auf das US-amerikanische Recht zugeschnitten ist, ist ihre

2 Free Software Foundation (Hrsg.): GNU Free Documentation License http://www.gnu.org/copyleft/fdl.html

Anwendung nicht ganz unproblematisch. Die inzwischen bessere Wahl wäre wahrscheinlich eine Creative-Commons-Lizenz, die von ihrer Zielsetzung über den Computer-Bereich hinausgeht und auch in national angepassten Versionen existiert. Derzeit arbeitet die Free Software Foundation[3] in Absprache mit der Creative-Commons-Initiative[4] an einer neuen Version der GFDL.

4. Der Neutrale Standpunkt

Die trotz des anarchistischen Charakters des Projekts notwendigen Regeln und Leitlinien für die Formulierung von Artikeln werden innerhalb der Wikipedia entwickelt. Sie etablieren sich weniger durch Autoritäten oder Mehrheitsentscheidungen, sondern dadurch, dass sich gewisse Praktiken als sinnvoll erweisen und deshalb von anderen Autoren aufgegriffen werden. Eine fundamentale Regel ist lediglich der so genannte „Neutrale Standpunkt".[5] Er verlangt, Ideen und Fakten in einer Weise zu präsentieren, dass sowohl Gegner als auch Befürworter einer solchen Idee deren Beschreibung akzeptieren können. Dies geschieht, indem gerade bei strittigen Themen alle wesentlichen Positionen und Argumente zu einem Thema angemessen erwähnt und ihren jeweiligen Vertretern zugeschrieben werden. Dem Ziel, eine für alle rational denkenden Beteiligten akzeptable Beschreibung zu formulieren, gehen oft lange Diskussionen und Streitigkeiten voraus – bis hin zu so genannten *EditWars*, bei denen zwei Benutzer in kurzer Folge ihre Änderungen rückgängig machen.

5. Vorbilder und Geschichte

Obwohl die Wikipedia ein grundsätzlich neues Konzept darstellt, lassen sich Vorbilder und Vorläufer festmachen. Die Präambel der Satzung des Vereins Wikimedia Deutschland beginnt selbstbewusst mit einem Zitat des berühmten Enzyklopädisten Denis Diderot: „. . . damit die Arbeit der vergangenen Jahrhunderte nicht nutzlos für die kommenden Jahrhunderte gewesen sei; damit unsere Enkel nicht nur gebildeter, sondern gleichzeitig auch tugendhafter und glücklicher werden, und damit wir nicht sterben, ohne uns um die Menschheit verdient gemacht zu haben." (Diderot 1969, S. 79) Die selbstgestellte Aufgabe, eine Enzyklopädie zu erstellen, misst sich weniger an herkömmlichen Standardwerken wie dem Brockhaus und der Encyclopædia Britannica – die natürlich dennoch gerne für einen Vergleich herangezogen werden –, sondern am Anspruch, das gesamte Wissen der Menschheit zu sammeln. Im Gegensatz zu den traditionellen Wissenssammlungen und Enzyklopädien geschieht dies konsequent als *Hypermedia*. Als moderne Vorläufer werden deshalb Vannevar Bushs Memex (1945) und Ted Nelsons Xanadu-Project (1960) genannt. Ein fast schon in Vergessenheit geratenes Projekt aus dem dokumentarischen Bereich sind die Arbeiten

3 http://www.gnu.org/fsf/fsf.html
4 Creative Commons International: Deutschland http://de.creativecommons.org/ [31. Okt 2004]
5 Bei der Wikipedia wird der „Neutrale Standpunkt" als „*natural point of view*" (npv) bezeichnet.

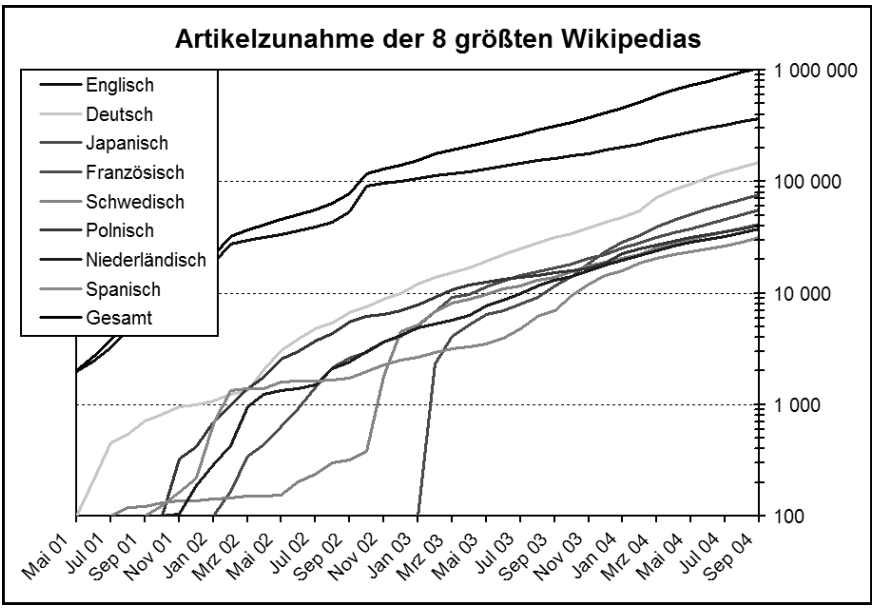


Abbildung 1: Ab einer bestimmten Größe wachsen die Wikipedias exponentiell (Statistik erstellt mit Zachte 2004)

Paul Otlets am *Répertoire bibliographique universel* (Boyd 1994, S. 235–250) für das *Office International de Bibliographie*, dessen Visionen der Wikipedia erstaunlich nahe kommen.

Als direkter Vorgänger der Wikipedia ist die im März 2000 gegründete Nupedia zu nennen, in der die Qualität der einzelnen von Wissenschaftlern freiwillig erstellten Artikel in einem aufwändigen Peer-Review-Prozess sichergestellt werden sollte. Da dies relativ schleppend voranging, gab es die Idee, die schon damals unter der GPL stehende Nupedia durch einen offeneren Bereich zu ergänzen.

Larry Sanger, Chefredakteur von Nupedia, und Jimmy Wales, der Gründer und Finanzier des Projekts, riefen zu diesem Zweck am 15. Januar 2001 das Wikipedia–Projekt ins Leben. Wikipedia entwickelte sich bald ungleich besser als die Nupedia selbst: Bereits nach einem Monat konnten mehr als 1 000 und Anfang September mehr als 10 000 Artikel verzeichnet werden. Inzwischen (2004) hat das Projekt die Millionen-Artikel-Marke überschritten. Dabei macht der englischsprachige Teil nur etwa die Hälfte aus und ist damit aber umfangreicher als alle bekannten Enzyklopädien zusammen genommen, da die Wikipedia auf viele miteinander verbundene unabhängige Sprachversionen ausgeweitet worden ist.

6. Gemeinschaftliches Arbeiten

Ende Juni 2003 wurde die Wikimedia Foundation als Stiftung nach dem Recht des US–Bundesstaates Florida gegründet. Sie wahrt die Namensrechte und betreibt die Server, auf denen die verschiedenen Wikimedia-Projekte laufen. Als deutsche Schwesterorganisation folgte ein Jahr später der als gemeinnützig anerkannte Verein „Wikimedia Deutschland – Gesellschaft zur Förderung Freien Wissens e. V." Die Organisationen nehmen keinen nennenswerten Einfluss auf die Inhalte, sondern kümmern sich vor allem um organisatorische Aufgaben wie Presse- und Öffentlichkeitsarbeit sowie das Sammeln von Spenden und Fördergeldern. Die anlaufenden Kosten für den Betrieb der Server (inzwischen mehr als 250 € pro Tag) werden ausschließlich durch Spenden gedeckt – Anfang Oktober kamen durch einen Aufruf innerhalb von 14 Tagen über 50 000 US-Dollar für den Kauf neuer Hardware zusammen.

Der Erfolg der Wikipedia ist nicht nur durch ihre Inhalte bedingt, sondern lässt sich auch dadurch erklären, dass sich unter den Beteiligten schnell eine Community gebildet hat. Über Mittel wie die Versionsgeschichten bekommt man schnell mit, wer an bestimmten Artikeln arbeitet und erhält dadurch Kontakt zu Autoren mit ähnlichen Interessen. Durch die intensive Zusammenarbeit und das gemeinsame Ziel, eine möglichst gute Enzyklopädie im Rahmen des „Neutralen Standpunkts" zu erstellen, wird ein Zusammengehörigkeitsgefühl hergestellt. Da die große Mehrzahl der Teilnehmer konstruktiv mitarbeitet, werden mutwillige Entstellungen von Artikeln (Vandalismus) schnell wieder entfernt.

Unter den Autoren bilden sich nicht nur Arbeitsgemeinschaften, sondern auch Spezialisierungen heraus. So kontrollieren einige Nutzer Texte auf Rechtschreibung, andere stellen sicher, dass die Artikel vernünftig mit anderen verlinkt sind und wieder andere ordnen die Artikel nur in die verschiedenen Kategorien ein. Viele Autoren beginnen mit solch kleinen Änderungen und entdecken auf diese Weise wie einfach es ist, sich produktiv zu beteiligen.

7. Die Artikel

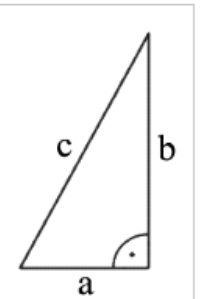
Die einzelnen Artikel der Wikipedia lassen sich unter URLs in der Form *http://de.wikipedia.org/wiki/Artikelname* aufrufen. Die Subdomain (hier 'de') entspricht bis auf einige Ausnahmen dem jeweiligen Sprachkürzel nach ISO-639. Beim Seitenaufruf wird entweder der entsprechende Artikel oder ein Hinweis angezeigt, der besagt, dass ein Artikel dieses Namens bisher nicht existiert, aber direkt erstellt werden kann.

Die Bearbeitung des Artikelinhalts oder einzelner Kapitel geschieht in einer speziellen Wiki-Syntax, die keine HTML-Kenntnisse erfordert und relativ schnell zu erlernen ist (siehe Abbildung 2). So trennen beispielsweise Leerzeilen einzelne Absätze, und URLs werden automatisch als Hyperlink dargestellt. Auch komplexere Auszeichnungen wie Listen und Tabellen sind möglich – bis hin zu mathematischen Formeln in LaTeX, Zeitleisten und sogar Hieroglyphen.

Satz des Pythagoras

aus Wikipedia, der freien Enzyklopädie

Der **Satz des Pythagoras** ist ein mathematischer Satz der Geometrie. Er besagt, dass einem rechtwinkligen Dreieck die Fläche des Quadrats über seiner Hyptenuse (in der Grafik als *c* bezeichnet) gleich der Summe der Flächen der Quadrate über seinen Katheten (*a* und *b*) ist.



Mathematisch wird dies folgendermaßen ausgedrückt:

Sind a, b, c *die Seiten eines rechtwinkligen Dreiecks mit* c *als Hypothenuse, so gilt*

$a^2 + b^2 = c^2$

Siehe auch: Fermatscher Satz

```
[[Bild:pythagoras_abc.png|right|Rechtwinkliges Dreieck]]

Der '''Satz des [[Pythagoras von Samos|Pythagoras]]''' ist ein
mathematischer Satz der [[Geometrie]]. Er besagt, dass in einem
[[Winkel|rechtwinkligen]] [[Dreieck]] die [[Fläche]] des [[Qua-
drat]]s über seiner [[Hypotenuse]] (in der Grafik als ''c'' be-
zeichnet) gleich der Summe der Flächen der Quadrate über seinen
[[Kathete]]n (''a'' und ''b'') ist.

Mathematisch wird dies folgendermaßen ausgedrückt:

:''Sind ''a, b, c'' die Seiten eines rechtwinkligen Dreiecks
mit ''c'' als Hypothenuse, so gilt''

::<math>a^2 + b^2 = c^2</math>

'''Siehe auch:''' [[Fermatscher Satz]]
```

Abbildung 2: Beispiel eines Artikels und seines Quelltextes in Wiki-Syntax

Neben den normalen enzyklopädischen Artikeln gibt es eine Reihe von Namensräumen, die durch einen Doppelpunkt getrennt den Artikelnamen vorangestellt werden. So existiert zu jedem Artikel eine parallele Diskussionsseite, beispielsweise „Diskussion:Satz des Pythagoras“ für den Artikel „Satz des Pythagoras“. Dort können Kommentare, Fragen zum Artikel und strittige Passagen diskutiert werden. Artikel über das Projekt selbst (Anleitungen und Projektorganisation) stehen in einem eigenen Wikipedia-Namensraum. Jeder angemeldete Benutzer besitzt eine öffentliche Benutzerseite, auf der er sich vorstellen und Notizen aufbewahren kann.

Durch die Referenzierung über eindeutige Titel bilden die Artikel der Wikipedia ein kontrolliertes Vokabular. Für synonyme Bezeichnungen kann ein so genannter

Redirect angelegt werden, der zur Vorzugsbenennung weiterleitet. Homonymen müssen durch Umformulierung oder Homonymzusätze unterschieden werden. Zusätzlich wird für jedes Homonym eine so genannte Begriffsklärungsseite angelegt. Von dort wird auf die einzelnen Begriffe verwiesen und mit einem Standardhinweis gebeten, Verweise auf das Lemma dahingehend zu ändern, dass sie direkt auf den jeweils passenden Artikel zeigen. Da Begriffsklärungsseiten keine Themen, sondern mögliche Bedeutungen einer Bezeichnung behandeln, lassen sie sich auch als eine Art von Wörterbucheinträgen auffassen, die ansonsten nicht in der Wikipedia gewünscht sind.

8. Mehr als normale Texte

Eine der wesentlichen Funktionen eines Wikis ist die Versionsgeschichte eines jeden Artikels. Jede Änderung wird protokolliert und kann bei Bedarf verbessert oder rückgängig gemacht werden. Die letzten Änderungen in der gesamten Wikipedia lassen sich über eine Liste einsehen und so durch andere Wikipedianer überprüfen. Zusätzlich können angemeldete Benutzer Artikel auf ihre Beobachtungsliste setzen, so dass sie über Änderungen an diesen informiert werden. Aus der Versionsgeschichte eines Artikels können zwei beliebige Versionen ausgewählt und miteinander verglichen werden (siehe Abbildungen 3 und 4).

Als Hypertexte sind die Artikel der Wikipedia hochgradig untereinander verlinkt. Dies wird auch als einer der Gründe dafür angesehen, warum die Wikipedia in den Ergebnislisten von Suchmaschinen wie Google relativ hoch platziert wird. Zum Anlegen eines Verweises genügt es, den Titel des zu referenzierenden Artikels in doppelte eckige Klammern zu setzen (siehe Beispielartikel in Abbildung 2). Als Linktext kann auch eine andere Zeichenkette gewählt werden (im Beispiel „rechtwinkligen" für einen Link auf den Artikel „Winkel"). Verweise auf bisher nicht existierende Artikel erscheinen automatisch rot hervorgehoben. Klickt man auf einen dieser Links, wird man aufgefordert, den entsprechenden Artikel selbst zu beginnen.

Da alle Verweise zwischen Wikipedia-Artikeln in einer Datenbank gespeichert werden, lässt sich direkt ermitteln, von welchen Artikeln auf einen anderen verwiesen wird („Links auf diese Seite") und welche noch nicht existierenden Artikel häufig verlinkt werden („Gewünschte Artikel"). Neben Links auf normale Artikel innerhalb einer Wikipedia kann auch auf Artikel des gleichen Themas in anderssprachigen Wikipedias verwiesen werden. Die Links auf andere Sprachen werden gesondert zu jedem Artikel angezeigt und fördern den Zusammenhalt zwischen den verschiedensprachigen Wikipedias. Dank der gemeinsamen Lizenzierung unter der GFDL können fremdsprachige Artikel übersetzt werden, so dass auch kleinere Wikipedias vom Wachstum der großen Sprachen profitieren.

Neben Links innerhalb des Textes werden oft unter „Siehe auch" assoziative Verweise auf andere relevante Artikel angelegt, deren Zusammenhang im Artikeltext noch nicht genügend erläutert wurde (siehe Beispielartikel in Abbildung 2). Am Ende eines Artikels folgen unter „Weblinks" Verweise auf andere Internetseiten und unter „Literatur" weiterführende Literaturangaben. Während sich unter den Weblinks meist viele weiterführende Quellen finden und bei einigen Artikeln sogar regelmäßig Spam–

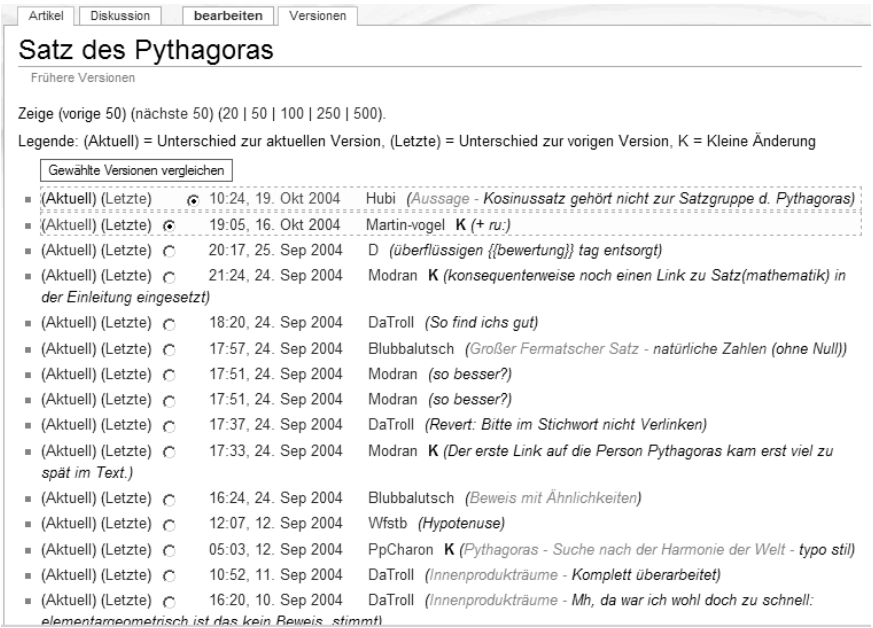


Abbildung 3: Versionsgeschichte eines Artikels

Abbildung 4: Vergleich zweier Artikelversionen

Einträge entfernt werden müssen, sieht es bei den Literatur- und Quellenangaben allerdings leider oft noch etwas spärlich aus. ISBN-Nummern im Text verweisen automatisch auf die „ISBN-Suche“[6], einem speziellen Artikel, der unter anderem eine Reihe von Buchhändlern und Bibliothekskatalogen auflistet. Die jeweilige ISBN-Nummer wird bei einer Auswahl an diese mit übermittelt. An Bibliothekskatalogen werden zur Zeit der Karlsruher Virtuelle Katalog (KVK) und die Kataloge des Gemeinsamen Bibliotheksverbund (GBV), des Kooperativen Bibliotheksverbund Berlin-Brandenburg (KOBV), des Hochschulbibliothekszentrum des Landes Nordrhein-Westfalen (HBZ) sowie des Österreichischen Bibliotheksverbund unterstützt.

Ein Schwachpunkt der Wikipedia ist sicherlich die mangelnde Recherchefunktion. Falls ein gesuchtes Lemma nicht bekannt oder vorhanden ist, fällt es manchmal schwer, die passenden Artikel zu einem Thema zu finden. Die Software bietet lediglich eine einfache Volltextsuche, die in der Vergangenheit regelmäßig wegen Überlastung der Server abgeschaltet wurde. Als eine Alternative bietet sich die indirekte Suche über Google oder Yahoo an. Wünschenswert wäre eine speziell auf den Datenbestand der Wikipedia zugeschnittene Suchfunktion, wie sie ansatzweise auf der ersten CD-Ausgabe realisiert worden ist.

Einen thematischen Einstieg bieten die Portale und Kategorien. Kategorien sind frei vergebbare Schlagwörter, die sowohl den Artikeln als auch anderen Kategorien zugeordnet werden können, so dass Thesaurusstrukturen entstehen können. Da der Software jedoch bislang grundlegende Funktionen zur Thesaurusverwaltung fehlen und die Vergabe von Kategorien ohne einheitliches Regelwerk stattfindet, wäre es übertrieben, von einer kontrollierten Verschlagwortung zu sprechen. Vielmehr handelt es sich um eine Form des *social tagging* (Lee 2004). Diese Form der gemeinschaftlichen Inhaltserschließung hat in den letzten Monaten vor allem in Weblogs und durch Bookmarking-Systeme wie *del.icio.us* an Auftrieb gewonnen. Wie weit diese sehr dynamischen und damit bislang relativ inkonsistenten Systeme eine kontrollierte Erschließung beispielsweise mit Klassifikationen wie der Dewey Decimal Classification (DDC) oder der Regensburger Verbund Klassifikation (RVK) ersetzen können, wird sich zeigen.

9. Qualität

> „Wie ihr seht, kann ich hier einfach so eure Seite bearbeiten. Achtung: ich mach das nur, um euch auf eure Sicherheitslücke hinzuweisen.“ [7]

Regelmäßig werden erstaunte E-Mails an *info@Wikipedia* geschickt, in denen davor gewarnt wird, dass alle Seiten direkt und von jedem verändert werden können. Diese gewollte Offenheit ist auch häufigster Kritikpunkt: Wie kann die Zuverlässigkeit der Informationen sichergestellt werden, wenn jeder diese verändern kann und wie stark kann man sich auf die Informationen aus der Wikipedia verlassen?

6 http://wikipedia.de/Spezial:Booksources

7 aus einer E-Mail an *info@wikipedia.de*

Grundsätzlich sind auch gedruckte Werke und herkömmliche Nachschlagewerke nicht frei von Fehlern. Die Notwendigkeit kritischen Lesens kann auch die Wikipedia nicht aufheben. Allgemein wird jedoch darauf vertraut, dass Texte, die gemeinsam von etlichen Leuten begutachtet und überarbeitet wurden, weniger Fehler enthalten als ein Text eines einzelnen Autors. Somit wird auf die Selbstheilungskräfte des „Wiki-Prinzips" vertraut, durch die über kurz oder lang irgendwann jeder Fehler von irgendjemandem entdeckt wird, der ihn dann sofort verbessern kann. Dies täuscht jedoch nicht darüber hinweg, dass fragwürdige Formulierungen in der Wikipedia oft über einen längeren Zeitraum bestehen bleiben.

Vor allem bei sehr speziellen Themen dauert es einige Zeit, bis ein anderer Fachkundiger den Artikel findet, ihn liest und auch korrigiert. Ein Großteil der Artikel ist daher noch sehr unvollständig, einseitig oder schlicht fehlerhaft, so dass der Vergleich der gesamten Wikipedia mit einer herkömmlichen Enzyklopädie gewagt ist. Andererseits braucht sich die Wikipedia nicht zu verstecken, wie unlängst Tests der Zeitschrift „c't" und der Zeitung „Die Zeit" gezeigt haben. Dort wurde die Qualität der Artikel im Vergleich zu Encarta und Brockhaus sogar mit „sehr gut" bewertet (Kurzidim 2004, Schult 2004).

Die Anzahl der Artikel entspricht ungefähr herkömmlichen Enzyklopädien, und teilweise werden diese hinsichtlich Umfang, Verständlichkeit und Aktualität sogar übertroffen. Von Tag zu Tag vergrößert sich dieser Teil. Ein Problem besteht jedoch darin, dass es für den Leser nicht auf den ersten Blick ersichtlich ist, ob er einen guten oder einen schlechten Artikel vor sich hat. Vor allem in Bereichen, die als besonders schwierig gelten, wie politische Themen[8] und Themen der Pseudowissenschaften, da hier Überzeugungen aufeinander treffen und sich hier nur schwer durch Diskussion ein „Neutraler Standpunkt" finden lässt. Je kontroverser und umfassender das Thema, desto länger dauert es, bis sich ein guter Artikel stabilisiert. In Zukunft ist geplant, Artikel, die sich auf einem hohen Niveau stabilisiert haben, mit einer Auszeichnung zu versehen, damit diese für den Leser erkennbar sind.

Während in der Presse ein im Großen und Ganzen durchweg positives Bild der Wikipedia vermittelt wird, sind die Wikipedianer selbst ihre schärfsten Kritiker. Auf der Mailingliste und den Diskussionsseiten finden regelmäßig hitzige Diskussionen über die tatsächliche Qualität der Artikel statt. Da alle Leser zum Finden und Verbessern von Fehlern aufgefordert sind, kennen regelmäßige Mitarbeiter die Schwachstellen erwartungsgemäß am besten. Darüber, ob und wie lange Unsinn unkorrigiert in Artikeln stehen bleiben kann, ob immer mehr Experten an das Projekt gebunden werden oder aber durch langwierige Diskussionen abgeschreckt werden und nicht zuletzt, ob die Artikel tatsächlich immer besser oder ab einem bestimmten Niveau auch wieder schlechter werden, gibt es aussagekräftige Untersuchungen bislang nur in Ansätzen.

8 Siehe beispielsweise taz vom 05. Oktober 2004 zur Diskussion des Wikipedia-Artikels „Waffen SS" – http://www.taz.de/pt/2004/10/05/a0150.nf/text.ges,1 [31. Okt 2004].

10. Aktionen zur Verbesserung der Qualität

Um die Verbesserung der Qualität zu forcieren, haben sich in der deutschsprachigen Wikipedia verschiedene Methoden etabliert. In der *Qualitätsoffensive* wird alle zwei Wochen ein bestimmtes Themengebiet herausgegriffen, um Lücken und Schwachstellen darin aufzudecken und die Artikel gemeinsam zu überarbeiten – in der Vergangenheit beispielsweise Themen wie „Antarktis", „Olympische Spiele" und „Schweiz". Hilfreich bei der Fehlersuche sind verschiedene manuell oder automatisch erstellte Listen wie die *Artikelwünsche* und die *Kurzen Artikel*.

Mit den so genannten *Bewertungsbausteinen* lassen sich einzelne Artikel hinsichtlich bestimmter Kriterien, wie beispielsweise Verständlichkeit, als mangelhaft kennzeichnen. Eine Auszeichnung für besonders gute Artikel ist die Aufnahme in die Liste der *Exzellenten Artikel*. Dazu vorgeschlagene Artikel werden zuvor einem kritischen Review-Prozess, an dem jeder teilnehmen kann, unterzogen. Als Aushängeschild der Wikipedia wird wöchentlich ein exzellenter Artikel auf der Startseite präsentiert. Weitere Aktionen zur Verbesserung der Qualität sind unter anderem ein Schreibwettbewerb und der überwachte Einsatz von Programmen (sog. *Bots*), die die Wikipedia automatisch beispielsweise nach Rechtschreibfehlern durchforsten.

Auf einen längeren Zeitraum als die Qualitätsoffensiven sind die mittlerweile über 120 *Portale* und gut halb so viele *Wikiprojekte* angelegt. In ihnen werden Artikel zu einem Gebiet wie beispielsweise Religion, Mathematik, Feuerwehr, Astronomie & Raumfahrt oder Graphentheorie zusammengefasst. Teilweise werden dabei auch Bewertungen von Artikeln vorgenommen, die beispielsweise zu kurz oder besonders herausragend sind. Einem herkömmlichen Lektorat kommt die Erstellung eines *WikiReaders*[9] am nächsten. Hierbei werden für ein Thema die wesentlichsten Artikel gesammelt und einheitlich für den Druck gesetzt. Bisher sind zwei WikiReader zum Thema Internet und Schweden auf Papier und weitere WikiReader (z. B. Portugal und Wale) als PDF erhältlich.

11. Zusammenarbeit mit anderen Informationsanbietern

Für die Verwaltung stark strukturierter, normierter oder durch Relationen in Bezug gesetzer Informationen eignen sich Datenbanken oft besser als normale Wikis, welche eher für textuelle Informationen konzipiert sind. Beispielsweise sind einheitlich erfasste Metadaten besser in einem Katalog bzw. einer Datenbank mit konkretem Datenmodell aufgehoben. Gleichzeitig sind viele dieser Informationen auch für die

9 WikiReader (ISSN der Online-Ausgabe: 1613-7752) ist eine unregelmäßig erscheinende Heftreihe, welche ausgewählte Wikipedia-Artikel thematisch bündelt und in einer redaktionell aufbereiteten Form präsentiert. Die Auswahl der Artikel erhebt keinen Anspruch auf Vollständigkeit, sondern soll gewissermaßen als „Schnappschuss" des jeweiligen Themas dienen. Wir ermuntern unsere Leser ausdrücklich dazu, selbst weiter zu recherchieren, Artikel in der Wikipedia zu verbessern oder auch neue Artikel hinzuzufügen und damit Anregungen zu liefern für zukünftige WikiReader-Ausgaben. Ziel dieser Heftreihe ist es, einerseits eine kostengünstige Materialsammlung zu verschiedensten Themenbereichen zur Verfügung zu stellen (beschränkt auf die Druckkosten) und zum anderen, die Wikipedia auch „offline" bekannter zu machen.

Wikipedia von Interesse. So finden sich schon jetzt eine Vielzahl von Artikeln zu Personen, Büchern, Filmen, Firmen, Bands etc. Falls dazu bereits Datenbanken existieren, wäre eine Zusammenarbeit zwischen Wikipedia und Datenbankanbietern wünschenswert.

Aus Sicht der Wikipedia ist es natürlich das Ziel, alle diese Inhalte als *Open Content* frei zugänglich zu machen. Aber auch unabhängig davon ist eine Verlinkung untereinander machbar. Beispielsweise sollte von Wikipedia-Artikeln zu einzelnen Personen auf Nachweise zu ihren Werken in Nationalbibliographien, z. B. auf die *Internet Movie Database*[10] für Filme, auf *Freedb*[11] oder *MusicBrainz*[12] für Tonaufnahmen, auf *Kalliope*[13] für Nachlässe und so weiter, verwiesen werden. Hilfreich für die Verknüpfung wären Normdaten wie die PND – leider stehen diese aber in der Regel nicht frei als *Open Content* bzw. unter GFDL lizenziert zur Verfügung.

Für die Anreicherung und das Verlinken von Wikipedia-Artikeln und anderen Wikimedia-Projekten mit strukturierten Daten gibt es verschiedene Vorschläge. Diese reichen von der automatischen Erzeugung von Wiki-Artikeln aus Datenbankinhalten über die Verwendung von Formatvorlagen bis hin zu einer kompletten Wiki-Datenbank. Dafür schlägt Erik Möller unter http://meta.wikimedia.org/wiki/Wikidata vor, die MediaWiki-Software so zu erweitern, dass beliebige strukturierte Daten nach dem Wiki-Modell eingegeben und bearbeitet werden können.

12. Das Wissen der Welt

Die Zielsetzung von Wikimedia geht über die Erstellung einer Enzyklopädie hinaus und zielt darauf ab, „das Wissen der Welt zu sammeln und zugänglich zu machen." Dazu sind neben der Wikipedia inzwischen eine Reihe von weiteren Projekten im Rahmen der Wikimedia Foundation begonnen und eine noch größere Anzahl von Ideen für weitere Projekte zur gemeinschaftlichen Erstellung freien Wissens vorgeschlagen worden. Alle basieren auf freien Inhalten im Sinne der GFDL und Wiki-Technologien. Bereits benutzbar sind

- Wiktionary (Wörterbuch)
- Wikibooks (Lehr- und Fachbücher)
- Wikiquote (Zitate)
- Wikisource (Orginalquellen)
- Wikispecies (Artenverzeichnis)
- Wikimedia Commons (Multimedia, seit September 2004)
- Wikinews (Nachrichten, seit Dezember 2004)

10 http://www.imdb.com
11 http://www.freedb.org
12 http://www.musicbrainz.org
13 Kalliope bezeichnet eine Datenbank für Nachlässe und Autographen in Deutschland.

Allerdings beschränken sich die meisten davon bisher auf englischsprachige Inhalte und haben teilweise noch mit technischen Problemen zu kämpfen. Da die MediaWiki–Software ursprünglich nur für Wikipedia entwickelt wurde, müssen für stärker strukturierte Informationen noch einige Anpassungen und Erweiterungen vorgenommen werden. Für einige Projektideen, wie beispielsweise *Wikiversity* (multimediale Lehrmaterialien) und *Wikimaps* (geographische Informationen), sollen die Erweiterungen mit Hilfe von Förderanträgen realisiert werden. Das neueste Projekt ist *Wikinews*, das das Ziel hat, gemeinsam über Nachrichten aller Art von einem neutralen Standpunkt aus zu berichten und sie zusammenzufassen.

Weitere Projekte beziehen sich auf die Verbreitung von freien Inhalten in anderer Form als im Internet. Für die sogenannten WikiReader werden Artikel eines Themenbereiches zusammengestellt und als PDF oder gedruckt herausgegeben. Die in Zusammenarbeit mit dem Verlag *Directmedia Publishing* hergestellte CD-ROM–Ausgabe der deutschsprachigen Wikipedia enthält rund 130 000 Artikel vom Stand des 1. September 2004. Um mehr als die darauf vorhandenen rund 1 200 Bilder auf einen Datenträger zu bekommen, soll die nächste Ausgabe 2005 auf DVD erscheinen. Auch eine Papierausgabe der wichtigsten Artikel der englischsprachigen Wikipedia ist geplant.

Ob diese Schwesterprojekte ebenso erfolgreich wie die Wikipedia sein werden, bleibt abzuwarten. Zumindest letztere ist nach anfänglicher Skepsis bereits ein voller Erfolg. Es ist zu vermuten, dass sich das „Wiki-Prinzip" der einfachen, offenen Bearbeitung in Verbindung mit freien Inhalten auch auf andere Inhalte übertragen lässt. Sicher ist, dass die Möglichkeiten der Wikipedia und verwandter Projekte noch lange nicht ausgeschöpft sind. Wie und wohin sie sich in den nächsten Jahren weiter entwickeln werden, bleibt spannend.

Literaturverzeichnis

Boyd, Rayward, W. (1994), 'Visions of Xanadu: Paul Otlet (1868-1944) and Hypertext', *Journal of the American Society of Information Science* **45**(4).
http://alexia.lis.uiuc.edu/~wrayward/otlet/xanadu.htm [31. Okt 2004].

Cunningham, Ward; Leuf, B. (2001), *The Wiki Way – Collaboration and Sharing on the Internet*, Addison-Wesley, Boston, MA.

Diderot, D. (1969), *Enzyklopädie – Philosophische und politische Texte aus der Encyclopédie*, dtv, München.

Kurzidim, M. (2004), 'Wissenswettstreit', *C't* **21**, S. 132 ff.

Lee (2004), 'Can social tagging overcome barriers to content classification?', headshift Weblog http://www.headshift.com/archives/002085.cfm [31. Okt 2004].

Schult, T. J. (2004), 'Lernen vom Schinken in Scheiben – Was taugen die aktuellen Enzyklopädien auf CD-ROM und DVD? Ein Test', *Die Zeit* **43**. zeit.de
http://www.zeit.de/2004/43/C-Enzyklop_8adien-Test [31. Okt 2004].

Zachte, E. (2004), 'wikistat-Skript', http://members.chello.nl/epzachte/Wikipedia/Statistics/ [31. Okt 2004].

Kapitel 7

Open Innovations

„Technology combines the physical world with the social, the objective with the subjective, the machine with the man. . . .Ultimately, any understanding of technological knowledge must recognize the composite nature of that knowledge. Our technological creations carry with them the traces of both the engineer and the larger society."

– Robert Pool (1997): Beyond Engineering: How Society Shapes Technology, S. 15, New York und Oxford: Oxford University Press.

„Innovation“ – eine Spurensuche

ROBERT A. GEHRING

1. Der Innovationsbegriff

Der Gebrauch des Innovationsbegriffes ist in den letzten Jahren zu einer Alltagserscheinung geworden. Als „Innovationsoffensive“ und in der jüngst erhobenen Forderung, sie als Staatsziel in der Verfassung festzuschreiben (heise newsticker 2005*a*), hat es die *Innovation* sogar bis in die Verlautbarungen von Spitzenpolitikern und -wirtschaftsvertretern geschafft hat. Manchmal kann man sich des Eindrucks nicht erwehren, dass die Wortwahl bloß Modeerscheinung sei, wie man vor einigen Jahren eben bei jeder sich bietenden Gelegenheit „new economy“ zu sagen hatte, wollte man sich als an der Spitze des Forschritts stehend präsentieren. Was aber, wenn überhaupt, verbirgt sich hinter dem so häufig gebrauchten Innovationsbegriff?

Der *Innovation* möchte ich mich an dieser Stelle in einem „zweistufigen Verfahren“ nähern, zum einen von der sprachlichen Seite her, zum anderen von der inhaltlichen, indem das Verständnis des Begriffs in zwei Disziplinen – Ökonomik und Recht – in gebotener Kürze herausgearbeitet wird. Derart ausgestattet können die Leser sich dann mit der praktischen Relevanz des Themas anhand dreier Aufsätze auseinandersetzen, die in dieser Einleitung knapp vorgestellt werden.

1.1. Etymologie der *Innovation*

Der Innovationsbegriff, obschon nicht mehr ganz neu, wird in seinem Bedeutungsgehalt um so schwieriger, je länger man sich auf Spurensuche in der Literatur begibt. Ein umfangreiches etymologisches Wörterbuch der deutschen Sprache lässt auf *innig* sogleich *Innung* folgen und verzichtet auf *Innovation*. Mehr Erfolg hat man, wenn man die elektronischen Wörterbücher aus dem Hause Duden heranzieht. Dort finden sich dann im Universalwörterbuch eine Reihe von Bedeutungen:

1. a) (Soziol.) *geplante u. kontrollierte Veränderung, Neuerung in einem sozialen System durch Anwendung neuer Ideen u. Techniken:* politische I.; das Wachstum durch I. fördern;
   b) (bildungsspr.) *Einführung von etw. Neuem; Neuerung; Reform*
2. (Wirtsch.) *Realisierung einer neuartigen, fortschrittlichen Lösung für ein bestimmtes Problem, bes. die Einführung eines neuer Produkts od. die Anwendung eines neuen Verfahrens:* technische -en.

3. (Bot.) *(bei ausdauernden Pflanzen) jährliche Erneuerung eines Teils des Sprosssystems.*

Man erfährt außerdem, dass die *Innovation* etymologisch den spätlateinischen Wörtern „innovatio" (Erneuerung, Veränderung) und „innovare" (erneuern, verändern) verwandt ist.

Mit einem Blick ins *Collins Concise Dictionary* (1998, S. 666) kann man sogar den Zeitpunkt der Wortbildung von „innovate" ermitteln: im 16. Jahrhundert, also in der Renaissance. Erinnert man sich des längst vergessenen Geschichtsunterrichts, so wird man mit der Renaissance eine Phase großer sozialer Umbrüche, technologischer Durchbrüche und die Herausbildung eines neuen Menschenbildes – Johannes Gutenberg vielleicht als sein Archetyp – verbinden.

Und schließlich erfährt man aus einem soziologischen Wörterbuch noch, dass *Innovation* auch kriminell sein kann (Hillmann 1994, S. 373):

> „In der Theorie abweichenden Verhaltens bezeichnet I. ein Verhalten, das in der Zielsetzung mit den allg. anerkannten kulturellen Standards übereinstimmt, sich aber illegitimer Mittel bedient; z. B. das kriminelle Verhalten unterprivilegierter Ges.gruppen, die in Ermangelung entspr. Anfangschancen im Konkurrenzkampf um Aufstieg und Statusverbesserung in der Leistungsges. Bereicherungsdelikte begehen."

Träger der *Innovation* sind die Innovatoren, denen letztlich der soziale Fortschritt zu verdanken ist (a. a. O.).

In diesem Sinne weiß man jetzt zwar ungefähr, *was* man sich unter *Innovation* so alles vorzustellen hat, aber *wie* es eigentlich dazu kommt, und welche *Rolle* der *Innovation* zukommt, bleibt weitgehend offen.

Zwei Disziplinen, die sich auf unterschiedliche, aber verquickte Weise mit dem Thema *Innovation* befassen, sind Ökonomik und Recht. Deren Beiträge zur Debatte sollen kursorisch untersucht werden.

2. *Innovation*, ökonomische Theorie und Wirtschaftspolitik

Unter deutschen Ökonomen wurde der *Innovation* lange Zeit nur wenig Aufmerksamkeit zuteil. „Die allgemeine ökonomische Theorie", konnte man vor knapp zwanzig Jahren lesen, „geht von den konkreten ökonomischen Gegebenheiten aus und versucht, die Ursachen der beobachteten Phänomene aufzudecken." (Luckenbach 1986, S. 4) *Innovation* schien damals nicht dazu zu gehören, fehlt doch im Index jeglicher Eintrag zu diesem Stichwort.

Ein Sprung vorwärts in der Geschichte der ökonomischen Theorie der Wirschaftspolitik lässt einen noch immer ratlos. Auch bei Woll (1992) gibt es die *Innovation* als solche nicht und – im Gegensatz zu „Bildungspolitik" und „Umweltpolitik" – anscheinend keinen Bedarf an „Innovationspolitik", „Technologiepolitik, „Forschungspolitik" usw.

Erst einige Jahre später findet die *Innovation* endgültig einen Platz in der Theorie der Wirtschaftspolitik: Fritsch et al. (2001) weisen im Index auf „Innovationsanreiz"

und „Innovationsverbreitungsfunktion" hin, wobei die Anreize deutlich mehr Platz einnehmen. Ebenso ist für Schmidt (1999) *Innovation* kein „terra incognita" mehr.

Doch auch im englischsprachigen Raum wurde die *Innovation* lange Zeit sehr stiefmütterlich behandelt. Die immer noch vorherrschende neoklassische Theorie der Ökonomie, wie sie prominent von der „Chicago School"[1] vertreten wird, tut sich schwer mit der *Innovation*. Der starre theoretische Rahmen der Preistheorie, früher als „general equilibrium theory" und heutzutage überwiegend als „Mikroökonomie" firmierend, bot und bietet wenig Platz für innovative Ansätze.

Peter F. Drucker hat es vor einigen Jahren sehr schön auf den Punkt gebracht (Drucker 1993, S. 26):

> „Classical economics optimizes what already exists, as does mainstream economic theory to this day, including the Keynesians, the Friedmanites, and the Supply-siders. It focuses on getting the most out of existing resources and aims at establishing equilibrium. It cannot handle the entrepreneur but consigns him to the shadowy realm of "external forces," together with climate and weather, government and politics, pestilence and war, but also technology. The traditional economist, regardless of school or "ism," does not deny, of course, that these external forces exist or that they matter. But they are not part of his world, not accounted for in his model, his equations, or his predictions."

Technologischer Wandel (J. Schumpeter), Transaktionskosten (R. Coase), Wirtschaftspsychologie (H. Simon), Institutionenökonomik (D. North) usw. usf. sind bis heute nicht in nennenswertem Umfang in den Kanon aufgenommen worden. Wo Gleichgewichte herrschen, ist für *creative destruction* (Schumpeter) eben kein Platz. Konsequenterweise sind die Beiträge der genannten Forschungsrichtungen zum Verständnis von *Innovation* kaum reflektiert worden. Die wenigen Erklärungsansätze fallen in Umfang und Gehalt äußerst dürftig aus und fußen letztlich auf Argumentationen aus dem 19. Jahrhundert, deren hauptsächlicher Mangel darin besteht, dass sie empirischer Prüfung nur ausnahmsweise – und nicht in der Regel – standhalten. Doch wir wollen erneut einen Blick in die Bücher werfen, um die Kerngedanken der momentan in der Ökonomik herrschenden Auffassung von *Innovation* herauszuarbeiten.

Beginnen wir mit zwei Standardwerken, Carlton und Perloff (2000) und Waldman und Jensen (2001).

Carlton und Perloff behandeln die Frage des technischen Fortschritts in einem eigenen Kapitel, unter der Überschrift „Patents and Technological Change" (S. 501–542). Sie fassen im Wesentlichen den Stand der unter den (neo-)klassischen Ökonomen herrschenden Meinung dahingehend zusammen, dass:

– Anreize in Form von Patenten, Urheberrechten und Markennamen notwendig seien (S. 501–507);

1 Eine moderate Einführung in die Bedeutung der „Chicago School" findet sich bei http://cepa.newschool.edu/het/schools/chicago.htm [15. Feb 2005].

- Imitation die Forschung hemme (S. 507–510);
- Patente sowohl die Forschung stimulierten als auch der Verbreitung von Ideen dienten (S. 510–513);
- Forschung durch Kooperation von Unternehmen vorangetrieben würde (S. 513–519);
- alternativ der Staat Preise als Anreize ausloben könnte (S. 518–521).

Anschließend widmen sie sich den ökonomischen Details des Patentwesens, etwa welche die bestmögliche Patentlaufzeit und wie zu lizenzieren sei, bevor sie sich dem Einfluss von Patenten auf die Marktorganisation zuwenden. Zusammenfassend kann man sagen, dass Carlton und Perloff (2000) eine sehr konsequente Darstellung der herrschenden Meinung gewählt haben, mithin davon abweichenden Auffassungen keinen Platz einräumen.

Waldman und Jensen (2001) befassen sich in Kapitel 13 (S. 396–435) mit „Technological Change and Research and Development". Den Anfang macht eine kurze Darstellung der Schumpeter'schen Auffassung von „kreativer Zerstörung", in der Schumpeter in Form zweier längerer Exzerpte selbst zu Worte kommt, darunter seine berühmten Sätze über die Bedeutung der „kreativen Zerstörung":

> „The opening up of new markets, foreign or domestic, and the organizational development from the craft shop and factory to such concerns as U. S. Steel illustrate the same process of industrial mutation–if I may use that biological term–that incessantly revolutionizes the economic structure from *within*, incessantly destroying the old one, incessantly creating a new one. This process of Creative Destruction is the essential fact about capitalism. It is what capitalism consists in and what every capitalist concern has got to live in." Schumpeter (1942) zit. bei Waldman und Jensen (2001, S. 397)

Schumpeter war Anhänger eines dynamischen Modells vom Kapitalismus. Demzufolge gehe es im Kapitalismus nicht nur darum, „unter den Bedingungen gegebener Ressourcenausstattung, gegebenen technischen Wissens, gegebener Produkte und gegebener Präferenzstrukturen der Nachfrager" (Fritsch et al. 2001, S. 73) ein Gleichgewicht herbeizuführen, wie es im *statischen Modell der vollständigen Konkurrenz* angenommen wird. Vielmehr sei der technologische Wandel inhärent, seien damit verbundene Strukturänderungen unvermeidlich.

Ausgehend von der Berufung auf Schumpeter diskutieren Waldman und Jensen (2001) dann den Zusammenhang zwischen technischem Forschritt und „Market Structure, Firm Size" (S. 399–412). Anschließend fragen sie nach empirischer Evidenz, ob Schumpeters Thesen also in der Praxis bestätigt werden oder nicht (S. 412–418), und kommen zu dem Schluss, dass es Unterschiede je nach Industriesektor gebe.

Im nächsten Abschnitt wenden sich die Autoren dann der Ökonomie des Patentwesens zu, wobei sich die Darstellung nicht wesentlich von der in Carlton und Perloff (2000) unterscheidet.

Die Begründung der Notwendigkeit von Patenten wird – aus (neo-)klassischer Perspektive – sehr schön von Fritsch et al. (2001, S. 73 f.) gegeben:

Unternehmen fehlte unter Bedingungen der vollständigen Konkurrenz der Innovationsanreiz, da Gewinne aus Effizienzsteigerungen, wie sie durch Einsatz innovativer Produktionsprozesse oder den Absatz innovativer Produkte resultieren würden, den Innovatoren nicht zugute kämen. Das läge daran, dass konkurrierende Unternehmen mit „unendlicher Reaktionsgeschwindigkeit" (S. 74) imitieren und somit den Wettbewerbsvorteil des Innovators zunichte machen würden. Wo keine Gewinne realisiert werden könnten, wäre es nicht möglich, Ausgaben für Forschung und Entwicklung (F&E) zu amortisieren, womit Verluste zu tragen wären. Nähme man beide Faktoren zusammen, entgangenen Gewinn aus Wettbewerbsvorsprung und fehlende Amortisation von F&E-Ausgaben, so fiele jeder Innovationsanreiz weg.

Im Sinne dieser Argumentation werden in der Literatur temporäre Monopole (in Form von Patenten) gerechtfertigt, welche die fehlenden Innovationsanreize durch staatlichen Eingriff in den Wettbewerb herstellen sollen. Die damit einhergehende Einschränkung von Wettbewerb und Berufsfreiheit sei im Interesse des höheren Gutes *Innovation* hinzunehmen. Gegen allzu starke Wettbewerbsverzerrungen, etwa durch Monopolmissbrauch, wäre mithilfe des Kartellrechts vorzugehen.

Ein Blick in ein etwas aktuelleres, spezifisch der Mikroökonomie gewidmetes Werk soll unsere kleine Stippvisite in die ökonomische Theorie der *Innovation* abrunden.

Salvatore (2003, S. 211) stellt fest, „[t]he introdution of innovations is the single most important determinant of a firm's long-term competitiveness . . ."[2] Firmen stehen demnach im Zentrum des Innovationsgeschehens, wobei zwischen Produkt und Prozessinnovationen zu unterscheiden sei. Ein wichtiger Hinweis auf die Realität von Innovationsprozessen folgt,[3] wenn der Autor betont (S. 211):

> „Contrary to popular belief, most innovations are incremental; that is, they involve more or less continous small improvements in products or processes rather than a single, major technological breakthrough. Furthermore, most innovations involve the commercial utilization of ideas that may have been around for years."

Patente werden bei Salvatore (2003) nur sehr kurz und am Rande behandelt. Er räumt ihnen deutlich weniger Platz ein als die vorangehend besprochenen Werke.

Aufschlussreich ist Salvatores Charakterisierung von *Innovation* als des wichtigsten Mittels eines Unternehmens, im Wettbewerb zu bestehen. Damit steht er deutlich in der Nachfolge von Schumpeter. Folgt man seiner Einschätzung, so ergibt sich daraus die interessante Frage, warum denn überhaupt gesonderte Anreize (siehe Abschnitt 3.) zu schaffen sind oder ob Gewinnstreben unter Wettbewerbsbedingungen nicht hinreichend sein müsste, um permanente *Innovation* zu erzwingen. Denn schließlich

2 Allerdings verzeichnet auch hier der Index *Innovation* nicht als eigenen Stichpunkt.

3 Über das konkrete *Wie* des Innovationsprozesses macht der Autor allerdings praktisch keine Ausführungen.

müssten sich im Wettbewerb ja die effizientesten Innovatoren durchsetzen, sollte man meinen.[4]

3. Innovation und „geistiges Eigentum“

Das Recht befasst sich mit *Innovation* nur indirekt, nämlich insofern es – im Sinne von Adam Smith – Rahmenbedingungen setzt, innerhalb derer die Akteure ökonomischen Aktivitäten nachgehen. Nicht deren Aufgabe sei es, dabei an und für sich moralisch zu handeln, sondern die Rahmenbedingungen müssten Anreize zum moralischen Handeln geben.[5] Bezieht man in den Moralbegriff die Wahrung gesamtgesellschaftlicher Interessen ein, so hat *in concreto* auch das Recht die Aufgabe, Anreize zur Balancierung von Eigeninteresse und gesamtgesellschaftlichen Interessen zu setzen. „Law is policy“ – Recht ist Politik –, heißt es deshalb bei den angloamerikanischen Realisten unter den Juristen. Recht, das Rahmenbedingungen für *Innovation* setzt, ist in diesem Sinne „Innovationspolitik“.

Welche Position vertritt nun die herrschende Meinung der deutschen Juristen zu *Innovation* und dem Schutz des „geistigen Eigentums“? Und was haben ihre amerikanischen Kollegen zum Thema beizutragen?

Schwerpunktmäßig ist bei der Diskussion zwischen Urheberrecht und Patentrecht zu unterscheiden. Ersteres reguliert, wie mit Kunstwerken und Computerprogrammen zu verfahren ist, Letzteres den Umgang mit „Erfindungen“. Es ist sicher sinnvoll, den Schwerpunkt der Betrachtung auf die Frage des Patentschutzes zu legen, weil hier die höchsten Hürden für bestimmte Innovationsprozesse (inkrementelle, kumulative) zu nehmen sind.[6] Im Unterschied zum Urheberrecht bietet nämlich im Patentrecht die unabhängige Schöpfung (Entwicklung) eines Schutzgegenstandes keine Verteidigungsmöglichkeit gegen die Ansprüche des ersten Rechtsinhabers. Das heißt, dass der Patentinhaber die Nutzung einer unabhängig entwickelten, dem Gehalt nach aber hinreichend ähnlichen Invention untersagen kann.

4 Vgl. Shapiro und Varian (1999) für eine umfangreiche Diskussion der Faktoren, die Innovationen und ihre Durchsetzung im Markt beeinflussen; vgl. hingegen Rivette und Kline (2000) für eine affirmative Darstellung von Patenten als „the “smart” bombs of tomorrow’s business wars.“ Nicht nur Wissenschaftler, auch Industrieverbände warnen davor, dass „over-protection may inhibit “follow-on” innovation by the millions who come after the initial creator.“ (Digital Connections Council of the Committee for Economic Development 2004, S. 2)

5 Oft vergessen, war Adam Smith nicht nur der Begründer der modernen Volkswirtschaftslehre, sondern auch ein anerkannter Moralphilosoph. Seine beiden Hauptwerke, „The Wealth of Nations“ und „The Theory of Moral Sentiments“, befinden sich – liebevoll aufbereitet – vollständig im Internet: http://www.econlib.org/library/Smith/smWN0.html und http://www.econlib.org/library/Smith/smMS0.html.

6 Im Bereich des Urheberrechts für Software stellt sich die Situation ähnlich dar. Und aus dem Beitrag von Bessen und Maskin geht hervor, dass im Zuge der sich ausbreitenden Internetnutzung zunehmend auch andere urheberrechtsrelevante Gebiete betroffen sind.

3.1. Patentschutz und Innovation

Um vielverbreiteten Missverständnissen entgegenzuwirken, ist eine genaue Unterscheidung hinsichtlich der Schutzgegenstände in Bezug auf unser Innovationsthema angebracht.

Ausgangspunkt ist eine Idee, die als solche (noch) nicht schutzfähig ist. Die Idee muss zuerst in eine deskriptive Form gebracht werden. Je nachdem, welche Form der Idee gegeben wird und welchem Zweck die Form dienen kann, wird es sich entweder um Kunst handeln, oder Aspekte der industriellen Anwendbarkeit und Herstellung überwiegen. Mit Ausnahme von Software werden industriell relevante Ideen ausschließlich vom Patentrecht erfasst, künstlerische Ideen vom Urheberrecht.

Software nimmt eine Position zwischen den Stühlen ein, da sie der Form nach dem Urheberrecht zugeordnet wird, der Funktion nach aber dem Patentrecht. In der Folge besteht ein in der Praxis relevanter Doppelschutz.

Eine Idee mit industrieller Anwendbarkeit, in deskriptive Form gebracht und die Anforderungen des Patentrechts erfüllend (d. h. einer erfinderischen Tätigkeit entsprungen, neu und gewerblich anwendbar), wird zur Erfindung (Invention). Um aber zu einer *Innovation* zu werden, muss aus der formalisierten Idee soziale Praxis werden (vgl. Tuomi 2002, S. 4), d. h. die Idee muss in gegenständlicher oder prozeduraler Form Verbreitung finden. Während das Patentrecht den ersten Schritt, von der Idee zur Erfindung, zum Gegenstand hat, wird der zweite Schritt, von der Erfindung zur *Innovation*, nicht erfasst. Die Überführung in die soziale Praxis ist nicht Voraussetzung, um ein Patent erhalten zu können. Das Patentrecht dient also dem Erfindungsschutz, nicht dem Innovationsschutz (vgl. Keukenschrijver et al. 1999, S. 17, Rn. 67). Analog dient das Urheberrecht dem „Schutz des Urhebers in seiner Beziehung zum Werk" (vgl. § 1 Urheberrechtsgesetz), nicht der Förderung der Verbreitung neuer Werke.[7]

Das Patentrecht schützt Erfindungen, wobei der Begriff der *Erfindung* im Gesetz selbst nicht definiert ist. Vielmehr gibt es auf der einen Seite gesetzliche Ausschlussbedingungen, d. h. Listen mit Gegenständen, die keine Erfindung sein sollen, und auf der anderen Seite (bei den Patentämtern) einen Kriterienkatalog, der bei der Beurteilung, ob ein Gegenstand denn nun eine Erfindung sei, abzuarbeiten ist. Man merkt schon an der hier gegebenen Darstellung, dass es keine scharfe Abgrenzung für Erfindungen gibt.[8] Das war in Deutschland von Anbeginn nicht unumstritten, und viele Kommen-

7 Im angloamerikanischen System war das historisch anders angelegt. Sowohl das erste „offizielle" Copyright-Gesetz, das Statue of Anne von 1710 (8. Anne c. 19) als auch die Copyright-Klausel der U. S.-Verfassung von 1787 stellten den Rechtsschutz als Mittel zum Zweck der Förderung von Wissenschaft, Technik und Lernen dar. Im Verlauf der letzten Jahrzehnte hat diese Philosophie aber an Bedeutung verloren, und als neuer Leitgedanke tritt der Investitionsschutz für Informationsgüter in den Vordergrund, der in der Legitimierung von Systemen für das Digital Rights Management seinen vorläufigen Kulminationspunkt erreicht hat (vgl. z. B. Bechtold 2002).

8 Seligsohn (1901, S. 27) erläutert, warum es einen trennscharfen Erfindungsbegriff nicht geben könne: „Die Ursache hiervon ist der zu definirende Begriff selbst, denn die Erfindung im patentrechtlichen Sinne lässt sich überhaupt nicht so scharf abgrenzen, daß man an der Hand ihrer Begriffsbestimmung ohne weiteres entscheiden könnte, ob im einzelnen Falle eine Erfindung vorliegt oder nicht. Es handelt sich hier um die Abschätzung, des Produkts einer Geistesarbeit, insbesondere um die Abwägung, ob dieses Produkt gegenüber dem bereits Vorhandenen einen so erheblichen technischen Fortschritt

tatoren versuchten sich daran, eine präzisere Begriffsdefinition für *Erfindung* zu geben. Hier zwei ausgewählte Versuche, zit. nach Seligsohn (1901, S. 27):

> „[. . .] Entdeckung einer vorher noch nicht bekannten Thatsache, daß durch eine konkrete technische Einwirkung auf einen Stoff der Außenwelt ein der Wiederholung an sich unterziehbarer Erfolg erzeugt wird [. . .]“ (Gareis)

> „[. . .] Ermittelung eines Verfahrens, wodurch die Herstellung eines Gebrauchsgegenstandes mit weniger oder anderer als der bisher nothwendigen Arbeit oder eines bisher ganz oder theilweise nicht bekannten Gebrauchsgegenstandes ermöglicht wird [. . .]“ (Quenstedt)

Der letzte vielleicht besonders interessant, bringt er doch implizit ein ökonomisches Effizienzkriterium ein, wenn von „weniger [. . .] Arbeit“ die Rede ist. Hier wird *Innovation* – nicht bloß Invention – angesprochen. Allerdings muss man hinzufügen, dass es sich um eine Ausnahmeerscheinung handelt, die folgenlos blieb. Moderne (deutsche) Kommentatoren betonen ausdrücklich, dass *Innovation* im Sinne ökonomischer Effizienzsteigerung nicht im Mittelpunkt des Patentrechtes stünde (Keukenschrijver et al. 1999, S. 17, Rn. 67):

> „Erfindungsschutz knüpft an die erfinderische Leistung, das Nichtnaheliegen, und nicht an Kategorien wie die “soziale Nützlichkeit” an. Maßgebend für den Patentschutz ist nicht die Bewertung der Folgen der Ausübung der Technik. [. . .] Als Wirtschaftslenkungsinstrument erscheint der Patentschutz a priori ungeeignet.“

„Law isn’t policy!“, könnte man diese Position wohl umschreiben, und sie macht das Dilemma im Verhältnis von Patentrecht und *Innovation* sehr schön deutlich: Geschützt wird *Invention*, nicht *Innovation*. Dabei stützt sich das Patentrecht[9] auf eine Reihe von Argumenten (vgl. Keukenschrijver et al. 1999, S. 15–16):

Eigentumstheorie Im Sinne der Naturrechtsauffassung wird eine geistige Schöpfung dem Schöpfer wie materielles Eigentum zugeordnet.

---

bedeutet, daß es sich rechtfertigt, seinem Urheber ein die allgemeine Gewerbefreiheit in dem Maße einschränkendes Recht, wie es der Patentschutz ist, zu gewähren. Denn daß dieser Schutz nicht jeder Neuerung zu Theil werden soll, sondern nur derjenigen, welche über die stetigen Fortschritte der Technik durch ihren geistigen Inhalt und ihre technische Wirkung hinausragt, darüber besteht keine Verschiedenheit der Meinung.“ Wenn man sich die hier vorgenommene Abgrenzung einmal genauer durch den Kopf gehen lässt, so hat es den Anschein, dass „inkrementelle“ Verbesserungen vor etwas mehr als hundert Jahren nicht für patentwürdig erachtet wurden.

9 Um präzise zu sein, müsste man zwischen dem kontinentaleuropäischen Patentrecht mit seiner naturrechtlichen Tradition und dem angloamerikanisch geprägten Patentrecht mit seiner Zweckmäßigkeitsbegründung unterscheiden. Bedingt durch die internationalen Entwicklungen, sind in den letzten Jahren allerdings starke Harmonisierungsbestrebungen erfolgreich gewesen, so dass die dogmatischen Unterschiede inzwischen deutlich in den Hintergrund getreten sind. Zu diesem Schluss kommt auch eine kürzlich vorgestellte EU-Studie (vgl. heise newsticker 2005*b*).

Belohnungstheorie Ein Erfinder, der seine Erfindung zum Patent anmeldet, legt diese zur allgemeinen Belehrung offen. Zur Belohnung dafür bekommt er ein exklusives Nutzungsrecht an der Erfindung erteilt.

Anspornungstheorie Das in Aussicht gestellte exklusive Nutzungsrecht mit den daran geknüpften Gewinnerwartungen soll zu inventiver Tätigkeit motivieren.

Offenbarungstheorie Durch die Offenlegung erteilter Patente verliert der Erfinder die Exklusivität des offenbarten Wissens. Im Gegenzug kann die Öffentlichkeit aus Patentschriften neues Wissen entnehmen. Um dem Erfinder für seine Offenheit eine Kompensation zu gewähren, erhält er für begrenzte Zeit ein exklusives Nutzungsrecht an der offenbarten Erfindung.

Man sieht, eine stringente Begründung für den Patentschutz gibt es nicht, vielmehr eine Art „Puzzle" aus den unterschiedlichsten Motiven.[10] Und weil das tradierte System des Patentschutzes eine solche amalgamierte Konzeption hat, provoziert es in der Praxis erhebliche Spannungen. Nicht nur, dass es aus Sicht innovativer Firmen unhandlich und oft schlicht irrelevant ist;[11] die durch den Patentschutz erzwungenen Transaktionskosten[12] wirken nachgewiesenermaßen kontraproduktiv, wenn es um schrittweise bzw. kumulative *Innovation* geht. Dieser Widerspruch ist dem Patentwesen immanent, besteht dessen Aufgabe doch zum Teil genau darin, die Weiterverbreitung von Innovationen einzuschränken, indem einem Erfinder exklusive Nutzungsrechte zugesprochen werden. Damit wird der soziale Nutzen zugunsten des individuellen Nutzens (des Erfinders) automatisch eingeschränkt.

3.2. Law & Economics

Die in den USA an den Hochschulen lehrenden Juristen vertreten mehrheitlich eine Denkrichtung, die sich „Law and Economics" auf die Fahnen geschrieben hat.[13] Den Grund für den Siegeszug ökonomischen Denkens in der Jurisprudenz bringen Cooter und Ulen (2004, S. 3 f.) in ihrem Standardlehrbuch auf den Punkt:

> „Economics provided a scientific theory to predict the effects of legal sanctions on behavior. [. . .] Generalizing, we can say that economics provides a behavioral theory to predict how people respond to changes in laws. This theory surpasses intuition, just as science surpasses common sense."

10 Für eine detaillierte Diskussion der verschiedenen Begründungsansätze vgl. Andersen (2004).

11 Regelmäßig weisen empirische Studien nach, dass Patente als Schutzinstrumente für Innovationen aus Sicht innovativer Unternehmen wenig attraktiv sind. Vgl. u. a. Gallini (2002), Blind et al. (2003) und die Beiträge in OECD (1996), besonders Kapitel 5.

12 Zum Beispiel für Lizenzverhandlungen, wenn jemand eine Folgeinnovation produzieren will, zu der er nicht das ursprüngliche Patent hält.

13 Die deutsche Spielart davon, die sich der „ökonomischen Analyse des Rechts" verschrieben hat, weist allerdings mehr Unterschiede als Gemeinsamkeiten damit auf. Schon die Disparitäten in den Bezeichnungen verweisen auf die stark differierenden Priorität. Ökonomik findet unter den Rechtsgelehrten hierzulande wenig Zuspruch – Ausnahmen bestätigen die Regel. Ein Vergleich der Lehrbücher bestätigt diesen Eindruck.

So gut diese Sätze auch klingen mögen, hilft uns das Buch von Cooter und Ulen (2004) auf der Suche nach der *Innovation* aber nicht weiter. Ein Blick in den Index zeigt, dass *Innovation* für die Autoren kein Thema zu sein scheint. Allerdings widmen sie im Rahmen der Behandlung von Eigentumsrechten auch dem „geistigen Eigentum" mehrere Seiten (S. 122–37). Im weitgehenden Einklang mit ihren deutschen Kollegen wird dem Patentrecht *inventionsförderliche* – nicht *innovationsförderliche* – Wirkung zugesprochen. Sie betonen jedoch, dass man selbst dafür bis heute keine wissenschaftlichen Belege hat erbringen können (S. 123):

> „Intellectual property law, however, is a historical accretion that developed without a scientific basis. Only recently has property law come under economic analysis. Even today, however, available economic analysis is insufficient to the task. The usual technique of economic analysis involves comparing equilibria [. . .] whereas intellectual property law requires an analysis of innovation and changing technology [. . .]"

Besonders hervorzuheben ist ihre nüchterne Einschätzung, dass sich die Rechtslage im Bereich des „geistigen Eigentums" im Wesentlichen nach „politically powerful special interest groups who care about their own profits more than the nation's wealth" richten würde.

Andere Lehrbücher wie Dwyer und Menell (1998), Merges et al. (2000), Lemley et al. (2003) oder *das* Standardwerk zu Softwarepatenten, Stobbs (2000) helfen auf der Suche nach dem Zusammenhang zwischen „geistigem Eigentum" als ökonomischem Instrument zur Förderung der *Innovation* auch nicht weiter. Ein letztes, oben schon zitiertes Werk, verbleibt zur Prüfung: Landes und Posner (2003).

Das Buch der beiden Vertreter der „Chicago School" behandelt ausdrücklich „The Economic Structure of Intellectual Property Law", wie der Titel schon verrät. Wo, wenn nicht hier, sollte man die gesuchte Aufklärung erfahren . . . ?

> „The standard rationale of patent law is that it is an efficient method of enabling the benefits of research and development to be internalized, thus promoting innovation and technological progress." (Landes und Posner 2003, S. 294)

Die Autoren fügen dem sogleich hinzu, dass sie eine differenzierte Betrachtungsweise[14] für angebracht halten, die sie in Kapitel 11 ihres Buches auch im Detail durchführen. Ihre sehr lesenswerte Behandlung des Themas beginnt mit der Abgrenzung vom Copyright (S. 294), setzt sich mit formalen ökonomischen Betrachtungen – Modell, Diagramm und Formeln – fort (S. 296–300), um dann zu wettbewerblichen und Wohlfahrtsüberlegungen überzugehen. Alle ökonomischen Analysen halten sich im Rahmen der neoklassischen Preistheorie auf, was sie einerseits in die herrschende

14 Was Landes und Posner hier als die gängige Erklärung für die Funktion von Patenten wiedergeben, das fällt sofort ins Auge, steht den Auffassungen deutscher Juristen in scharfem Gegensatz gegenüber.

ökonomische Literatur integriert, andererseits auch deren Schwächen voll zum Tragen kommen lässt. Je mehr sie sich prozeduralen Fragen des Patentwesens nähern (S. 300-310), desto skeptischer wird der Ton ihrer Ausführungen.

Unter der Überschrift „Is Patent Law Socially Cost-Justified?" wenden sich Landes und Posner dann ab S. 310 den neuralgischen Punkten des existierenden Patentwesens zu. Schon anhand der Menge und des Inhalts der Fußnoten kann der Leser sich des Eindrucks nicht erwehren, dass eine qualifizierte Mehrheit von Autoren den aktuellen Zustand des Patentwesens mehr als zweifelhaft findet. Landes und Posner selbst kommen zumindest zu keinem klaren Schluss, der eine positive ökonomische Bewertung der bestehenden Rechtslage darstellen würde. Als Ausweg stellen sie schließlich Patente als Antwort auf durch Geschäftsgeheimnisse und Monopole aufgeworfene Probleme dar (S. 326–323). Was der Leser bei ihnen nicht finden kann, ist eine Antwort auf die Frage nach der *Innovation*. Im ganzen Kapitel wird weder erklärt, wie Innovationen zustandekommen, noch wie „geistiges Eigentum" zwecks Innovationsförderung am besten gestaltet sein sollte.

4. Kurzkritik der herrschenden Meinung

An dieser Stelle soll die mehr als umfangreiche Kritik zur herrschenden Meinung unter Ökonomen und Juristen nur kurz elaboriert werden, um das Auge des Lesers für die Bedeutung der nachfolgenden Beiträge zu schärfen.[15]

– Betrachtet man die herrschenden ökonomischen Theorien und die von ihnen präferierten Schutzinstrumente des Immaterialgüterrechts, Urheber- und Patentrecht, zusammen, wird offensichtlich, dass es darin weitestgehend nicht um *Innovation* geht. *Innovation* ist weder eindeutig kausal mit den Schutzinstrumenten in Zusammenhang zu bringen,[16] noch dient sie als Ziel der Modellbildung.[17]

15 Um dem eventuellen Vorwurf der Einseitigkeit der Darstellung etwas entgegenzusetzen, möchte ich an dieser Stelle ausdrücklich betonen, dass es eine Fülle an aktuellen Werken zu Innovationsthemen gibt, die eine andere bzw. differenziertere Auffassung als die herrschende Meinung vertreten. Vgl. u. a. allgemein Tuomi (2002), Foray (2004); speziell zum Patentwesen Jaffe und Lerner (2004). Landes und Posner (2003, S. 310), beide revolutionärer Gesinnung sicher unverdächtig, stellen ausdrücklich fest, dass es keinen Beleg für die Effizienz des Patentschutzes gäbe: „The most important economic question about the patent system is whether on balance [. . .] it increases or reduces economic welfare. Although there are powerful economic reasons in favor of creating property rights in inventions, there are also considerable social costs and whether the benefits exceed the costs is impossible to answer with confidence on the basis of present knowledge." Doch weder haben diese anderen Auffassungen es bisher in den Kanon des ökonomischen Lehrwissens geschafft, noch lässt sich ein nennenswerter Einfluss auf die Wirtschaftspolitik – national wie international – erkennen. Im Gegenteil: Statt die Bedeutung der Vielfalt der Innovationsprozesse zu berücksichtigen, findet in den letzten Jahren auf breiter Front – man denke etwa an das TRIPS-Abkommen, diverse Richtlinien der EU, Gerichtsurteile und die Aktivitäten der Patentämter – die Umsetzung einer sehr engen Vorstellung von *Innovation* statt, in der Patente und Urheberrechte als dominierende Instrumente der Innovationsförderung installiert werden. Auffällig ist, dass es keine vergleichbaren Publikationen aus Deutschland gibt.

16 Das ist unter anderem dem Umstand geschuldet, dass es keine praktikable Möglichkeit gibt, die notwendigen Kosten einer *Innovation* zu ermitteln und zu vergüten (vgl. Scotchmer 1998).

17 Insofern trifft hier zu, was Coase (1974, S. 211) über das Beispiel des Leuchtturms in der ökonomischen

– Je stärker *Innovation* vom Einsatz wissenschaftlicher Methoden abhängig ist, desto problematischer wird die im Immaterialgüterrecht angelegte Verhinderung der Verbreitung von Wissen. Das hat methodische Gründe, werden doch dadurch die in der Wissenschaft geforderten Eigenschaften Falsifizierbarkeit und Verifizierbarkeit grundsätzlich in Frage gestellt.
– Wie z. B. Castells (2001, S. 34 f.) betont, gewinnt die Informationsverarbeitung zunehmend an Bedeutung. Typisch für Informationen in der Informationsgesellschaft ist ihre (Re-)Kombinierbarkeit und Rückkoppelbarkeit (gefördert durch die Digitalisierung). *Innovation* speist sich zu wesentlichen Teilen aus diesen Eigenschaften von Informationen, die zunehmend in inkrementellen und kumulativen Innovationsprozessen ausgenutzt werden. Damit entziehen sich Informationen einer bloß linearen Betrachtungsweise, wie sie sowohl der ökonomischen Modellvorstellung als auch dem juristischen Paradigma für „geistiges Eigentum" zu Grunde liegt.
– Die Fragmentierung der Anwenderbedürfnisse in einer globalisierten Wirtschaft macht den Erfolg von Innovationen von der aktiven Einbeziehung von Anwendern abhängig. Der Anwender spielt aber in den herrschenden ökonomischen Modellvorstellungen von *Innovation* keine aktive Rolle. Und auch die Rechtsschutzregime für „geistiges Eigentum" weisen dem Anwender lediglich eine Rolle als Konsument, nicht aber als Koproduzent zu. Den Realitäten – insbesondere im Internet – kann man so nicht mehr gerecht werden.
– „Almost all questions regarding intellectual property law are open." (Cooter und Ulen 2004, S. 123)

Die elementare Schwäche der herrschenden Theorien zu *Innovation* und Innovationsanreizen ist das Fehlen einer empirischen Basis als Grundlage. Ohne belastbares empirisches Fundament wird es einer Theorie notwendig an Wissenschaftlichkeit mangeln. Daraus abgeleitete Schlüsse werden in den seltensten Fällen richtig sein, meist schlicht irrelevant. Werden aus einer solchen metaphysischen Theorie aber sogar normative, also politische, Ansprüche abgeleitet, wird es riskant und möglicherweise auch teuer. Auf die Ökonomie der *Innovation* trifft zu, was Coase (1974, S. 210 f.) zur Ökonomie der Leuchttürme feststellte:

> „I think we should try to develop generalizations which would give us guidance as to how various activities should be best organized and financed. But such generalizations are not likely to be helpful unless they are derived from studies of how such activities are actually carried out within different institutional frameworks. Such studies would [. . .] lead to generalizations which have a solid base."

Theorie gesagt hat, dass es aus der Luft gegriffen sei. Auch *Innovation* hängt, ökonomisch betrachtet, in der Luft.

5. Die Beiträge

Das vorliegende Kapitel zu „Open Innovation" stellt drei sehr unterschiedliche Beiträge zusammen, die sich auf je eigene Weise damit befassen, welch vielfältige Gestalt Innovationprozesse *in praxi* annehmen können. Das reicht von neuen Wegen der wissenschaftlichen Publikation (Wurch) über Kooperationen von Musikliebhabern im Internet (Bessen und Maskin) – und diesmal sind nicht Peer-to-Peer-Netzwerke gemeint – bis hin zu Windsurfern und Open-Source-Entwicklern (von Hippel). Gemeinsam ist ihnen, dass sie zeigen, wie *Innovation* außerhalb des von der herrschenden Meinung dominierten Paradigmas stattfindet, teils intentional, teils akzidentiell.

Die ursprüngliche Fassung des ersten Beitrags wurde von zwei ausgewiesenen Kennern der Innovationsproblematik, James Bessen und Eric Maskin bereits Ende der 90er Jahre verfasst, und später mehrfach aktualisiert (so auch für dieses Buch). Sie zeigen den durch die explosionsartige Entwicklung des Internets und seinen neuen Kooperationsmöglichkeiten provozierten Konflikt mit traditionellen Vorstellungen des „geistigen Eigentums": Auf der einen Seite werden gänzlich neuartige Innovationsprozesse möglich, auf der anderen Seite stehen dem gewachsene und neu geschaffene Rechtsansprüche gegenüber. Am Beispiel der Entwicklung des *Online Guitar Archive* (OLGA) zeigen sie, wie Anwender – Gitarristen – Mehrwert durch kooperative *Innovation* schaffen. Dabei bauen sie auf einem Musikbestand auf, der gemäß den urheberrechtlichen Eigentumsbestimmungen von Musikfirmen als Privateigentum angesehen wird. Die freie Verfügbarmachung der so entstehenden Leistungen kollidiert mit privaten Verwertungsinteressen. Eine vergleichbare Kollision zeigen sie zwischen der Arbeitsweise von Softwareentwicklern und dem Drang von einigen Unternehmen zu Softwarepatenten. Die Autoren weisen nach, dass die traditionelle Logik der Innovationsanreize, wie sie eingangs geschildert worden ist, im Internet offensichtlich nicht mehr als innovationsförderlich angesehen werden kann. Sie plädieren für schwächere Rechte des „geistigen Eigentums", um *Innovation* nicht zu behindern.

Der zweite Beitrag stammt aus der Tastatur eines Studenten der Informatik, Sören Wurch. Im Rahmen seiner Diplomarbeit setzt er sich mit der Problematik des wissenschaftlichen Publizierens auseinander. Er zeigt anhand beeindruckender Zahlen, wie das ökonomische Interesse an der Verwertung wissenschaftlicher Informationen und Erkenntnisse die Wissenschaft zu strangulieren droht. Kommerzielle Großverlage sind, und dabei stützen sie sich auf das Urheberrecht, nun mal primär an Profit orientiert. Die Verbreitung neuen Wissens – Voraussetzung für *Innovation* – ist ihnen nur Mittel zum Zweck. Bei zunehmender Konzentration im Verlagsgeschäft gelang es ihnen in der Vergangenheit, exorbitante Preissteigerungen bei den Bibliotheken, durchzusetzen. Jetzt, wo deren Finanzlage alles andere als rosig ist, werden Abonnements abbestellt, und den Wissenschaftlern bricht der Wissensnachschub zusammen. Die ersten unter ihnen sind nicht länger gewillt, das hinzunehmen. Sie suchen nach Auswegen, und unter dem Motto „Open Access" formiert sich eine Bewegung, die den Wissenschaftlern die Wissenschaft zurückgeben will.

Dass „Open Access" ein guter Weg sein kann, die Schaffung neuen Wissens und neuer Wissensprodukte zu fördern, haben die Entwickler von freier und Open–

Source-Software nun seit Jahrzehnten vorgemacht. Eric von Hippel, bekannter Innovationsforscher des MIT, beschließt Kapitel und Buch mit einem Rück- und Ausblick auf so unterschiedliche Phänome kooperativer *Innovation* unter Anwendern wie Hochleistungswindsurfen und Open-Source-Softwareentwicklung. Er zeigt, unter welchen Bedingungen es die Anwender sind, die Innovationsprozesse maßgeblich vorantreiben – „Hersteller entbehrlich". Folgt man den Überlegungen von Hippels, dann wird das in der Informationsgesellschaft immer häufiger der Fall sein. Wer den Zug zu „Open Innovation" verpasst, wird als Hersteller in absehbarer Zeit erkennen müssen, dass er entbehrlich geworden ist.

6. Paradigmenwandel: „Open Innovation"

In seinem berühmten Aufsatz „The Nature of the Firm" von 1937 fragte Ronald Coase, was es denn sei, das eine Firma zur Firma im Markt mache. Diese Frage beantwortete er mit der Feststellung, dass es die Transaktionskosten seien: Die Kooperation zwischen den eine Firma konstituierenden Personen sei kostengünstiger als es eine Organisation der Produktion nur über den Markt sein könne. Die Grenzen einer Firma seien durch ihre relativen Kostenvorteile bestimmt. Im Umkehrschluss verlieren diese Grenzen an Bedeutung, wenn die Kostenvorteile gegenüber anderen Formen der Kooperation sinken. In der Informationsökonomie – oder Wissensökonomie, was immer man bevorzugt – verschieben sich die Kostenstrukturen für Produktion, Beschaffung, Verarbeitung und Vertrieb. Das Internet leistet dazu einen entscheidenden Beitrag: Je höher der Anteil an Information bzw. Wissen ist, der zur Wertschöpfung beiträgt, desto geringer fällt der Kostenvorteil innerhalb einer Firma gegenüber dem Markt aus. Mehr noch, es werden Transaktionen jenseits der Firma und des Marktes möglich, sogar kostengünstiger, die sich in die traditionellen Vorstellungen der industriellen Organisation – so, wie sie in den Büchern beschrieben wird – nicht mehr ohne weiteres einordnen lassen. Das gilt erst recht für Innovationsprozesse. Sah man noch in der zweiten Hälfte des 19. Jahrhunderts, zur Zeit der Schaffung von Urheber- und Patentrecht, große Firmen als den zu schützenden Hort der *Innovation* an, sieht man sich nun mit einer neuen Realität konfrontiert. Die Grenzen der Firma werden durchlässig, wenn kooperative *Innovation* aus Kostengründen zum Paradigma wird. Forschung und Entwicklung lassen sich zukünftig nicht mehr in der fast hermetisch abgeriegelten Welt firmeneigener Labore durchführen. Die Zeit der Erfindungsfabriken ist vorbei. „Open innovation draws on technologies from networks of universities, startups, suppliers, and even competitors." (Chesbrough 2003*a*, S. 2) Es wird Zeit, dass sich auch die Gesetzgeber der neuen Situation bewusst werden und Innovationshemmnisse in der Form eines veralteten Urheber- und Patentrechts aus dem Weg räumen. Unternehmen, Wissenschaftler und Anwender sind schon längst dabei.

Literaturverzeichnis

Andersen, B. (2004), 'If 'Intellectual Property Rights' is the Answer, What is the Question? Revisiting the Patent Controversies', *Economics of Innovation and New Technology* **13**(5), S. 417–442.

Bechtold, S. (2002), *Vom Urheber- zum Informationsrecht. Implikationen des Digital Rights Management*, C.H. Beck, München.

Blind, K., Edler, J., Nack, R. und Straus, J. (2003), *Softwarepatente: Eine empirische Analyse aus ökonomischer und juristischer Perspektive*, Physica-Verlag, Heidelberg.

Carlton, D. W. und Perloff, J. M. (2000), *Modern Industrial Organization*, 3. Aufl., Addison Wesley Longman, Boston, MA.

Castells, M. (2001), *Das Informationszeitalter I: Die Netzwerkgesellschaft*, Leske + Budrich, Opladen.

Chesbrough, H. W. (2003*a*), 'Reinventing R & D Through Open Innovation', strategy+business enews. http://www.strategy-business.com/enewsarticle/?art=44635032 [1. Mai 2003].

Coase, R. H. (1937), 'The Nature of the Firm', *Economica* (4), S. 386–405. Wiederabdruck in R. H. Coase (1988): 'The Firm, the Market, and the Law.' S. 33–55.

Coase, R. H. (1974), 'The Lighthouse in Economics', *The Journal of Law and Economics* **17**(2), S. 357–376. Wiederabdruck in R. H. Coase (1988): 'The Firm, the Market, and the Law.' S. 187–213.

Collins (1998), *Collins Concise Dictionary*, 3. Aufl., HarperCollins, London, Glasgow und Sydney.

Cooter, R. und Ulen, T. (2004), *Law & Economics*, 4. Aufl., Pearson Addison Wesley, Boston, MA.

Digital Connections Council of the Committee for Economic Development (2004), 'Promoting Innovation and Economic Growth: The Special Problem of Digital Intellectual Property', http://www.ced.org/projects/ecom.shtml [04. Feb 2005].

Drucker, P. F. (1993), *Innovation and Entrepreneurship*, Harper Business, New York, NY.

Dwyer, J. P. und Menell, P. S. (1998), *Property Law and Policy: A Comparative Institutional Perspective*, The Foundation Press, Inc., Westbury, NY.

Foray, D. (2004), *The Economics of Knowledge*, The MIT Press, Cambridge, MA & London.

Fritsch, M., Wein, T. und Ewers, H.-J. (2001), *Marktversagen und Wirtschaftspolitik*, 4. Aufl., Franz Vahlen, München.

Gallini, N. T. (2002), 'The Economics of Patents: Lessons from Recent U.S. Patent Reform', *Journal of Economic Perspectives* **16**(2), S. 131–154.

Hillmann, K.-H. (Hrsg.) (1994), *Wörterbuch der Soziologie*, 4. Aufl., Alfred Kröner, Stuttgart.

Jaffe, A. B. und Lerner, J. (2004), *Innovation and its Discontents: How Our Broken Patent System Is Endangering Innovation and Progress, and What To Do About It*, Princeton University Press, Princeton, NJ & Oxford.

Keukenschrijver, A., Schwendy, K. und Baumgärtner, T. (Hrsg.) (1999), *Busse: Patentgesetz. Kommentar*, 5. Aufl., Walter de Gruyter, Berlin, New York.

Landes, W. M. und Posner, R. A. (2003), *The Economic Structure of Intellectual Property Law*, The Belknap Press of Harvard University Press, Cambridge, MA & London.

Lemley, M. A., Menell, P. S., Merges, R. P. und Samuelson, P. (2003), *Software and Internet Law*, 2. Aufl., Aspen Publishers, New York, NY.

Luckenbach, H. (1986), *Theoretische Grundlagen der Wirtschaftspolitik*, Franz Vahlen, München.

Merges, R. P., Menell, P. S. und Lemley, M. A. (2000), *Intellectual Property in the New Technological Age*, Aspen Law & Business, Gaithersburg, NY.

Newman, P. (Hrsg.) (1998), *The New Palgrave Dictionary of Economics and The Law (Vol. I–III)*, MacMillan Reference Limited, London.

OECD (1996), *Innovation, Patents and Technological Strategies*, OECD, Paris.

Rivette, K. G. und Kline, D. (2000), *Rembrandts in the Attic: Unlocking the Hidden Value of Patents*, Harvard Business School Press, Boston, MA.

Salvatore, D. (2003), *Microeconomics: Theory and Applications*, 4. Aufl., Oxford University Press, New York & Oxford.

Schmidt, I. (1999), *Wettbewerbspolitik und Kartellrecht*, 6. Aufl., Lucius und Lucius, Stuttgart.

Schumpeter, J. A. (1942), *Capitalism, Socialism, and Democracy*, Harper & Row, New York.

Scotchmer, S. (1998), *incentives to innovate*, Vol. 2 of Newman (1998), S. 273–277.

Seligsohn, A. (1901), *Patentgesetz und Gesetz, betreffend den Schutz von Gebrauchsmustern*, 2. Aufl., J. Guttentag, Berlin.

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

Stobbs, G. A. (2000), *Software Patents*, 2. Aufl., Aspen Law & Business, Gaithersburg & New York.

Tuomi, I. (2002), *Networks of Innovation: Change and Meaning in the Age of the Internet*, Oxford University Press, Oxford & New York.

Waldman, D. E. und Jensen, E. J. (2001), *Industrial Organization: Theory and Practice*, 2. Aufl., Addison Wesley Longman, Boston, MA.

Woll, A. (1992), *Wirtschaftspolitik*, 2. Aufl., Franz Vahlen, München.

heise newsticker (2005*a*), 'Bitkom fordert Verfassungsrang für Innovationen', http://www.heise.de/newsticker/meldung/56394, [15. Feb 2005].

heise newsticker (2005*b*), 'EU-Studie: Softwarepatentrichtlinie bringt amerikanische Verhältnisse', http://www.heise.de/newsticker/meldung/56493, [17. Feb 2005].

Geistiges Eigentum im Internet: Ist alte Weisheit ewig gültig?

James Bessen und Eric Maskin

Das Wachstum des Internets setzt die etablierten Formen des Schutzes geistigen Eigentums wie Urheber- und Patentrechte unter Druck. Informationen, einmal ins Internet gestellt, lassen sich leicht kopieren. Wo die Kopierkosten niedrig sind und die Anonymität gewahrt bleibt, reagieren die Rechteinhaber mit verschärfter Durchsetzung bestehender Rechtsansprüche und rufen nach dem Gesetzgeber. Doch der Ansatz, bestehende Rechtsansprüche zu verschärfen und neue zu schaffen, erweist sich im Internet als problematisch, wenn es darum geht, Innovation zu fördern. Im vorliegenden Aufsatz wird an Beispielen gezeigt, dass Anwender innovative Beiträge leisten, denen aber Urheberrecht und Patentrecht entgegenstehen. Die Autoren argumentieren, dass das Internet spezifische ökonomische Eigenschaften aufweist, die eine reflexartige Verschärfung von Ansprüchen auf geistiges Eigentum jedenfalls im Internet als ein ungeeignetes Instrument zur Innovationsförderung erscheinen lassen. Vielmehr gibt es empirische Evidenz dafür, dass schwächere Rechte des geistigen Eigentums die sequentielle Innovation besser unterstützen würden: Imitation erhöht in einer dynamischen Umgebung die Innovationsanreize; Lizenzierungskosten vermindern Innovationsanreize; reines Kopieren wäre zu verhindern, kreative Nachahmung hingegen zu fördern.*

1. Einleitung

Das Wachstum des Internets setzt die etablierten Formen des Schutzes geistigen Eigentums wie Urheber- und Patentrechte unter Druck. Informationen, einmal ins Internet gestellt, lassen sich leicht kopieren. Wo die Kopierkosten niedrig sind und die Anonymität gewahrt bleibt, reagieren die Rechteinhaber mit verschärfter Durchsetzung bestehender Rechtsansprüche und rufen nach dem Gesetzgeber, den Rechtsschutz auszuweiten: Zusätzlich zu den bestehenden Schutzrechten sollen neue Formen

* Der Text wurde von den Autoren für das vorliegende Buch aktualisiert. Die Übersetzung erfolgte mit Genehmigung seitens der Autoren durch Bastian Zimmermann, Clemens Brandt und Robert A. Gehring. Wir danken den Autoren für die Genehmigung zum Abdruck.

von „Inhalten", Medien und Zugriffsmöglichkeiten von Gesetzes wegen unter Schutz gestellt werden. Man kann diese Bemühungen als Teil einer seit 20 Jahren währenden Entwicklung sehen, die stets in Richtung Ausweitung von geistigen Eigentumsrechten und Intensivierung von deren Durchsetzung geht. Allerdings ist diese Reaktion nicht notwendigerweise angemessen, zumal im Internet.

In diesem Aufsatz wird argumentiert, dass das Internet und das darauf aufbauende World Wide Web spezifische Eigenschaften aufweisen, die einen solchen Ansatz – die permanente Verschärfung der Schutzrechte – als ungeeignet erscheinen lassen. Das Internet ist eine hochgradig dynamische und interaktive Gemeinschaft. Und tatsächlich ist ein großer Teil der Software, auf der das Web basiert, Open-Source-Software. Der vorliegende Aufsatz präsentiert in aller Kürze ein formales ökonomisches Modell zur Beschreibung einer solchen dynamischen und interaktiven Umgebung. Das Modell legt den Schluss nahe, dass unter den im Internet herrschenden Bedingungen sowohl die einzelnen Rechteinhaber als auch die Gesellschaft im Ganzen eher von *schwächeren* als von verschärften Schutzrechten profitieren würden.

Politiker sollten daher vorsichtig sein: Die überlieferte Ansicht, dass stärkere Schutzrechte für geistiges Eigentum automatisch die Innovationsanreize verstärken würden, basiert auf ökonomischen Modellvorstellungen, die dem Internet oft nicht gerecht werden.

2. Das traditionelle Modell von geistigem Eigentum

Üblicherweise wird strenger Schutz für geistiges Eigentum immer damit begründet, dass er für Autoren und Erfinder den Anreiz bewahrt, überhaupt etwas zu schaffen. Ausführlich lautet die Begründung folgendermaßen:

Kreative Tätigkeit birgt typischerweise erhebliche Kosten. Zwar sind Künstler, Autoren und Erfinder nicht unbedingt ausschließlich oder vorrangig von der Aussicht auf finanziellen Gewinn motiviert; dennoch kann der kreative Prozess, von der ursprünglichen Idee über ihre Entwicklung bis hin zur Verbreitung eines Werkes, so aufwendig sein, dass viele Urheber einen finanziellen Ertrag brauchen, um ihre Entwicklungskosten wieder zu erwirtschaften. Dieser Ertrag dient daher als „Innovationsanreiz".

- Wenn eine Arbeit kopiert wird, entgehen dem eigentlichen Autor/Erfinder mögliche Verkäufe, also Profite. Deshalb reduziert eine Umgebung, die das (kostenlose) Kopieren ermöglicht, den Innovationsanreiz. Mit einer geringeren Aussicht auf Profite werden manche Urheber die anfänglichen Entwicklungsinvestitionen nicht aufbringen können oder wollen. Sie werden sich daher gegen eine kreative Tätigkeit entscheiden.
- Der Rechtsschutz für geistiges Eigentum erschwert die Nachahmung und steuert so diesem Erosionseffekt entgegen. Der Schutz ermutigt Erfinder und Autoren in ihrem Schaffensprozess, wovon im Gegenzug die gesamte Gesellschaft profitiert.

In einem solchen Modell ist stärkerer Schutz automatisch besser: Stärkerer Schutz bewirkt eine Abnahme der Imitation, erhöht somit die Investitionsanreize und führt

letztendlich zu höherer Wohlfahrt. Diese Argumentation klingt überzeugend und ist seit über 200 Jahren grundlegend für einen starken Schutz des geistigen Eigentums.

Und doch ist das ökonomische Modell, das diesem traditionellen Argument zugrunde liegt, überraschend beschränkt. In Wahrheit ist kreative Tätigkeit oft nicht der Arbeit einsamer Schöpfer geschuldet. Vielmehr ist sie interaktiv und schließt Beiträge von vielen verschiedenen Seiten ein. Tatsächlich findet Innovation oft schrittweise statt. Jeder Innovationsprozess baut auf den Ergebnissen des vorangegangenen Schrittes auf, um eine Verbesserung zu erzielen. Das Standardmodell setzt Nachahmung oft mit Kopieren gleich. Wenn jedoch Innovation schrittweise stattfindet, ist Nachahmen mehr als Kopieren und stellt eine Bereicherung dar.

Das herkömmliche Modell basiert auf der Annahme eines einzelnen Schöpfers. Tief verankert in unserer Kultur ist das Bild des Kreativen als eines romantischen Individuums: der Künstler in der Dachstube oder der Erfinder in der Garage. Ein Teil der Überzeugungskraft des Standardmodells beruht auf unserer Angewohnheit, kreative Tätigkeit für das Spezialgebiet einsamer Genies zu halten.

Ein Ort, wo dieses vertraute Denken mit der Realität in Konflikt gerät, ist das World Wide Web. Das Web wird oft als Gemeinschaft bezeichnet. Es bietet den einzelnen Kreativen eine wunderbare Möglichkeit, ihre Werke zu veröffentlichen; zugleich offeriert es Möglichkeiten zur interaktiven Kommunikation. So bildet es den Nährboden für schrittweise Weiterentwicklungen.

Es ist hilfreich, einige Beispiele interaktiver und schrittweiser Innovationen im Web zu betrachten, um die Unzweckmäßigkeit des herkömmlichen Modells von geistigem Eigentum in dieser Umgebung besser zu verstehen.

3. Einige Beispiele für interaktive und schrittweise Innovation

3.1. Interaktive Foren

Ein bekanntes Beispiel für interaktive Entwicklung ist das *interaktive Forum*. Als Druckzeitschriften zunehmend im Netz veröffentlicht wurden, richteten ihre Herausgeber häufig auch sogenannte Foren ein. Diese interaktiven Webseiten bieten den Lesern bzw. der gesamten Öffentlichkeit die Möglichkeit, unabhängig Kommentare und Meinungen zu publizieren. Typischerweise kommen vielfältige Dialoge mit den Autoren von Artikeln zustande, die auch abgedruckt werden. Manche von ihnen nehmen die Form von Echtzeit-Gesprächen an, andere werden in E-Mail-Archiven gespeichert. Die Autoren steuern neues Material bei und diskutieren darüber mit den Lesern; die Leser wiederum liefern Feedback und bauen die Diskussion oft aus. Im Ergebnis entsteht eine stark erweiterte Version des Leserbriefs – mit sehr viel komplizierteren geistigen Eigentumsrechten.

Eine interaktive Website, auf der der Austausch „Unbedarfter“[1] zu Konflikten mit dem geistigen Eigentum geführt hat, ist das *Online Guitar Archive* (OLGA). OLGA wurde 1992 von James Bender gegründet und bietet ein Archiv mit etwa 28 000 von Nutzern eingestellten Gitarren-Tablaturen sowie Gitarrenübungen und andere

1 Das ist in Bezug auf das geistige Eigentum gemeint; Anm. d. Ü.

Unterstützung für Gitarristen. Die ehemals von der University of Nevada, Las Vegas, (UNLV) gehostete Seite war sehr beliebt: Nutzer luden sich von ihr etwa 200 000 Dateien pro Woche herunter.

Gitarren-Tablaturen sind eine Form der Musiknotation, die Bund- und Saiten-Fingersätze anzeigen. Die Tablaturen werden üblicherweise von Liedtexten begleitet. Da Gitarrenakkorde oft auf verschiedene Art gegriffen werden können, bieten die Tablaturen Anleitungen zu ihrer Umsetzung. Insbesondere helfen sie Gitarristen, die wie bestimmte Künstler auf CDs und anderen Aufnahmen klingen wollen. Für deren Spiel sind oft keine Noten verfügbar und selbst wenn, so treffen diese im Allgemeinen nicht genau auf die Aufführung auf der Aufnahme zu. Tablaturen werden üblicherweise von anderen Musikern als den ursprünglich spielenden notiert. Zwar werden sie mithilfe von Aufnahmen nach Gehör aufgeschrieben, aber verschiedene Musiker nehmen dieselbe Aufnahme unterschiedlich wahr, und kommerzielle Notenblätter neigen gern dazu, vereinfachte Fingersätze mit einem „hübschen" Klang zu präsentieren.

Die auf OLGA veröffentlichten Tablaturen stellen individuelle Interpretationen einzelner Gitarristen dar, die anhand von Aufnahmen ihre eigenen Fingersätze nieder geschrieben haben. Durch ihre Individualität fügen sie sowohl Aufnahmen als auch Notenblättern (so vorhanden) einen Mehrwert hinzu. Dessen ökonomische Bedeutung besteht darin, dass die Tablaturen eine Ergänzung zu den Audioaufnahmen darstellen, keinen Ersatz. Beide, der Verfasser der Tablatur und ihr Nutzer, der ein Lied einüben will, werden mit dem Anhören einer Aufnahme beginnen. Zwar mögen die Gitarren-Tablaturen gelegentlich Notenblätter ersetzen, aber insgesamt betrachtet stellen sie eine ökonomisch wertvolle Ergänzung zu Aufnahmen dar.

Dogmatisch betrachtet, stellt die Verbreitung von Gitarren-Tablaturen über das Internet wohl eine Verletzung traditioneller geistiger Eigentumsrechte dar. Die Plattenfirma EMI jedenfalls sah das so und schickte im Januar 1996 der UNLV einen Brief, in dem sie rechtliche Schritte androhte. Die UNLV schloss OLGA umgehend. Zahlreiche gespiegelte Server wurden ebenfalls vom Netz genommen. Einige gespiegelte Server blieben am Netz, jedoch mit verringerten Möglichkeiten, neue Tablaturen beizusteuern. Etliche Gitarristen im Web waren empört und begannen damit, EMI–Aufnahmen zu boykottieren. Für OLGA selbst – ohne rechtlichen Stand – war es nicht möglich, EMI zu Gesprächen über eine Verhandlungslösung zu bewegen. Im Jahr 1998 verschärfte sich die Situation noch, als die „Harry-Fox-Agentur" ins Spiel kam. Bei der Harry-Fox-Agentur handelt es sich um eine Organisation, die Musikaufnahmen lizenziert. Ohne Angabe konkreter Titel, die Urheberrechte verletzen würden, gab die Harry-Fox-Angentur dem OLGA sieben Tage Zeit, das komplette Archiv einschließlich gespiegelter Server zu schließen. Gespräche über mögliche Lizenzierungen wurden verweigert. OLGA konnte nur weiterbestehen, indem ein neues Archiv aufgebaut wurde, in das nur noch Gitarren-Akkordtabellen ohne komplette Liedtexte aufgenommen werden.

EMIs Handeln mag im Kontext der traditionellen Urheberrechtsvorstellung sinnvoll gewesen sein. In der Welt des Webs aber, in der Nutzer Mehrwert beisteuern, erscheint EMI kurzsichtig. Aus unserer Perspektive wäre es für EMI besser gewesen, das Hosting für die OLGA-Site zu übernehmen und die dort gespeicherten Tablaturen

mit dem eigenen Bestand an Aufnahmen, Fanclub-Informationen usw. zu verknüpfen. Andere Unternehmen geben in der Tat eine Menge Geld aus, um genau solche Dinge zu tun. Was wäre geschickter, als die Schirmherrschaft über eine ohnehin schon erfolgreiche Website zu übernehmen?

Im traditionellen Modell des geistigen Eigentums läßt sich die Wertschöpfung Außenstehender nicht integrieren. Damit eignet es sich nur schlecht als Leitfaden für Unternehmen oder Politik.

3.2. Sequentielle Entwicklung

Der Bereich der Publikation von Software bietet ein besonders anschauliches Beispiel für sequentielle Entwicklung, denn der Rechtsschutz für Software durchlief in den 80er Jahren eine Art natürliches ökonomisches Experiment. Zwar bildet die Verbreitung von Software im Web einen eigenständigen Bereich der Publikation, aber andere Publikationswege sind von Entwicklungen beim Schutz des geistigen Eigentums an Software in zweierlei Hinsicht betroffen.

Zum einen werden alltägliche Webaktivitäten von Patentinhabern potentiell bedroht, die ihre Rechte ausüben wollen könnten. Regelmäßig tauchen derartige Patente mit breiten Auswirkungen auf im Web übliche Handlungen auf, wie z. B. Comptons Multimedia-Patent, das Freeny-Patent auf elektronische Verkäufe und Unisys' GIF-Patent.

Zum anderen werden Inhalte häufig gemeinsam mit Software angeboten, wobei die Software den Inhalt überhaupt erst zugänglich macht. Beispiele dafür sind aktuelle Nachrichten, die zusätzlich zu den traditionellen Verbreitungsformen nun in Form von Datenbanken, Faxdiensten, E-Mail-Angeboten, Echtzeit-Meldungen und Radioberichten angeboten werden. Datenbankanbieter liefern oft auch einen Service, der an bestimmte Zugriffsarten gebunden ist, und heute stehen Technologien zur Verfügung, die einen technisch kontrollierten Zugang zu so ziemlich jedem Inhalt ermöglichen.

Die besondere Bedeutung von Software für unsere Diskussion liegt nun darin, dass Software typischerweise einer schnellen und schrittweisen Entwicklung unterworfen ist: Jede Innovation wird von Wettbewerbern imitiert und verbessert, wobei jeder einen einzigartigen Beitrag leistet. Softwarekonzepte, die in den 60er Jahren konzipiert wurden, wie das elektronische Publizieren, der Hypertext, Multimedia und künstliche Intelligenz, mussten erst über Jahrzehnte von vielen Firmen immer wieder verbessert werden, bevor sie kommerziellen Erfolg auf breiter Front hatten.

Die Veränderungen des Rechtsschutzes für Software starteten Mitte der 80er Jahre ein ökonomisches Experiment. Bis zu diesem Zeitpunkt war Software weitgehend durch das Urheberrecht geschützt und eine Reihe von Prozessen hatte dazu beigetragen, den Schutzumfang zu präzisieren. Es war klar, dass unmittelbares Kopieren illegal war, andere Formen der Nachahmung aber nicht, solange sie bei gleicher Funktionalität anders verwirklicht worden waren. Patente auf Softwareerfindungen wurden bis dahin nur ausnahmsweise von Gerichten bestätigt.

Ab Mitte der 80er und während der 90er Jahre wurde der Patentschutz für Software durch Gerichtsurteile erheblich ausgeweitet. Im Ergebnis gab es geradezu eine

Explosion der jährlich erteilen Softwarepatente auf über 20 000 (vgl. Abb. 1). Damit erreichten Softwarepatente einen Anteil von mehr als 15 % aller Patente (vgl. Abb. 2). Bis dato wurden in den USA mehr als 200 000 Softwarepatente erteilt. Dem traditionellen ökonomischen Modell zufolge hätte dieser entscheidend erweiterte Rechtsschutz für geistiges Eigentum die Innovationsanreize deutlich erhöhen müssen. Firmen, die aufgrund des Nachahmungsrisikos bisher davor zurückgeschreckt waren, hätten mit neuen Produkten in den Markt eintreten sollen. Projekte, deren Finanzierung bisher zu riskant erschien, hätten nunmehr realisierbar werden müssen. Es hätten in der Folge die Forschungs- und Entwicklungsausgaben steigen müssen.

Vielen aus der Software- und Computerbranche erschien die Ausweitung des Patentschutzes auf Software als ein Versuch, etwas zu reparieren, das gar nicht kaputt war. Diese Industriezweige waren doch bereits hochinnovativ: Neugründungen von Firmen, Produktinnovationen sowie Ausgaben für Forschung und Entwicklung waren im Verhältnis zu den Verkäufen zahlenmäßig hoch. Dennoch stand das hohe Innovationspotential nicht unbedingt im Widerspruch zum traditionellen Modell: Die innovativen Firmen hätten sozusagen bloß die Spitze eines Eisberges gebildet, wobei sie ausschließlich die profitabelsten Innovationen realisiert hätten, um in Anbetracht des Risikos der Nachahmung bestehen zu können.

Folgt man dieser Art der Argumentation, hätte ein stärkerer Rechtsschutz für geistiges Eigentum zu einem noch höheren Niveau an innovativer Aktivität führen müssen. Darüber hinaus hätte man erwarten können, dass stärkere Anreize mehr innovativen Start-up-Firmen den Eintritt in die jeweilige Branche erleichtern würden. In der Tat waren kleine Start-up-Firmen in der Vergangenheit eine wichtige Innovationsquelle in der Softwareindustrie.

Doch es geschah nichts dergleichen. Eine Untersuchung von Bessen und Hunt (2004) belegt detailliert, dass Softwarepatente größtenteils von großen, etablierten Unternehmen erworben wurden. Die meisten von ihnen sind im Hardwarebereich aktiv. Mögen sie auch Software in ihre Produkte integrieren (z. B. in Kopiergeräte), stellt sie dennoch nicht ihr eigentliches Produkt dar. Die kleinen Start-up-Firmen, die für die Innovation in der Softwareindustrie so wichtig waren, haben in der Wirklichkeit nur einen sehr geringen „Hang zu Patenten". Daher können Softwarepatente diesen Firmen keine stärkeren Innovationsanreize geboten haben. Noch wichtiger ist, dass Bessen und Hunt (2004) zu dem Schluss kommen, dass diejenigen Firmen, die während der 90er vergleichsweise viele Softwarepatente erhielten, ihre Forschungs- und Entwicklungsausgaben im Vergleich zu den Verkaufszahlen sogar reduzierten.

Es ist klar, dass diese Resultate nur schwer mit dem traditionellen Modell in Einklang zu bringen sind. Es scheint ganz so, dass (a.) Softwarepatente weitgehend auf der Basis von Forschung und Entwicklung erlangt wurden, die sowieso stattgefunden hätten, und dass (b.) die Erweiterung des Patentschutzes sowohl auf Forschungs- und Entwicklungsausgaben als auch auf Firmenneugründungen keinen sehr positiven Einfluss hatte. Nachahmungen scheinen Innovationen in einer dynamischen Umgebung mit rapider und schrittweiser Innovation nicht zu verhindern. Hier spielen Faktoren eine Rolle, die über das traditionelle Modell hinausgehen.

4. Eine kurze ökonomische Analyse

Ein passenderes ökonomisches Modell muss Folgendes berücksichtigen:

1. Kreative Nachahmung unterscheidet sich vom Kopieren; Nachahmer können für wichtige Bereicherungen sorgen.

2. Einige Umgebungen sind statisch, andere dagegen sind von hoher Dynamik und schrittweiser Innovation gekennzeichnet. Geistiges Eigentum kann in jeder dieser Umgebungen eine sehr unterschiedliche Rolle spielen.

3. Manche kreative Arbeiten haben einzelne Autoren, andere haben mehrere, aufeinander folgende.

4. Der Beitrag zusätzlicher Autoren ist oft nicht vorherzusagen, und der Wert jedes Beitrags besteht oft in der Beisteuerung des spezifischen Wissens des jeweiligen Autors.

Ein solches formales Modell zu entwickeln, geht über den Rahmen dieses Aufsatzes hinaus. Jedoch lassen sich einige bedeutende Ergebnisse eines solchen Modells, wie es an anderer Stelle dargestellt wurde (Bessen und Maskin 2000), folgendermaßen zusammenfassen:

1. Nachahmung kann in einer dynamischen Umgebung die Gesamtanreize erhöhen, etwas Neues zu schaffen. Der Großteil kreativer Tätigkeit stellt sich in der Tat als teilweise Nachahmung dar.

2. Starker Rechtsschutz für geistiges Eigentum kann Schaffensanreize dadurch vermindern, dass er Lizenzierungen und andere Formen gemeinsamer Informationsnutzung reduziert.

3. Im Großen und Ganzen ist ein gemäßigter Schutz geistigen Eigentums optimal. Die beste Art von Rechtsschutz für geistiges Eigentum ist stark genug, um unmittelbares Kopieren und geschäftszerstörende Plagiate zu verhindern, aber auch schwach genug, um die wechselseitige Lizenzvergabe zu maximieren und zur intensiven gemeinsamen Informationsnutzung zwischen Wettbewerbern zu ermutigen.

Die ersten drei Jahrzehnte der Halbleiterindustrie geben ein Beispiel dafür, wie Schutz für geistiges Eigentum in einer dynamischen Umgebung gut funktioniert. Angefangen mit den „Bell Labs", die ihre einfachen Patente auf den Transistor an jeden Interessenten gegen eine geringe Gebühr lizenzierten – obwohl es sich dabei um einen möglichen Wettbewerber handeln konnte –, lizenzierten Halbleiterunternehmen allgemein ganze Patentportfolios. Patente verhinderten nicht, dass neue Unternehmen den Markt betraten, und Unternehmen teilten sich wichtige Patente. Dieses Umfeld veränderte sich schließlich mit der in den frühen 80er Jahren vollzogenen, erheblichen Verschärfung des Rechtsschutzes für geistiges Eigentum.

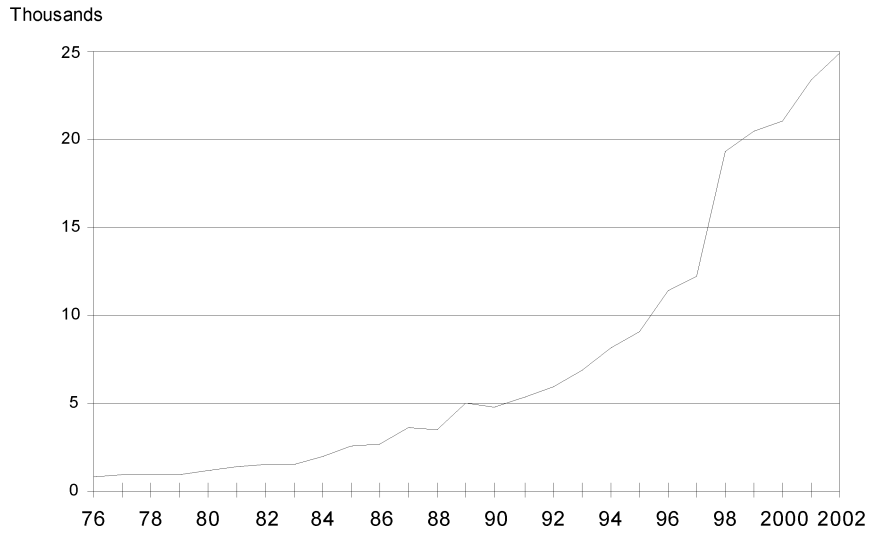


Abbildung 1: Gewährte Softwarepatente in 1 000 (Das Datum bezieht sich auf die Erteilung, nicht die Beantragung des Patents)

Freie und Open-Source-Software sind ebenfalls von dieser Dynamik sequentieller und komplementärer Innovation erfasst. Freie und Open-Source-Lizenzen ermöglichen Imitation und komplementäre Erweiterungen. Einige der Lizenzen verlangen, solche Weiterentwicklungen zu teilen, andere ermutigen lediglich dazu. Im Kern ist es aber das Teilen, das die Dynamik der Entwicklungen bestimmt.

5. Zusammenfassung

Wer Informationen veröffentlicht, ist gut beraten, den Wert der Beiträge anderer zu erkennen und sie zu solchen zu ermutigen. Die Gesellschaft als Ganzes würde davon profitieren anzuerkennen, dass neue Generationen von Autoren bedeutende Beiträge zu vorhandenen Werken leisten. Der Schutz des geistigen Eigentums sollte auf das Maß beschränkt werden, das notwendig ist, um den Ausgleich zwischen der Verhinderung des einfachen Kopierens und der Förderung schöpferischen Imitierens herzustellen.

Gelegentlich wird die Politik des geistigen Eigentums als ein Balanceakt beschrieben, bei dem es darum geht, den Schutz der Anreize zur Innovation gegen die gesellschaftlichen Vorteile einer Weiterverbreitung neuer Ideen abzuwägen. Die hier vorgelegte Analyse macht deutlich, dass sich die Politik nicht nur allgemein der (etwas strukturlosen) Verbreitung von Ideen als solcher widmen sollte, sondern auch den konkreten Formen der Imitation zum Zwecke der Verbesserung, wie sie insbesondere

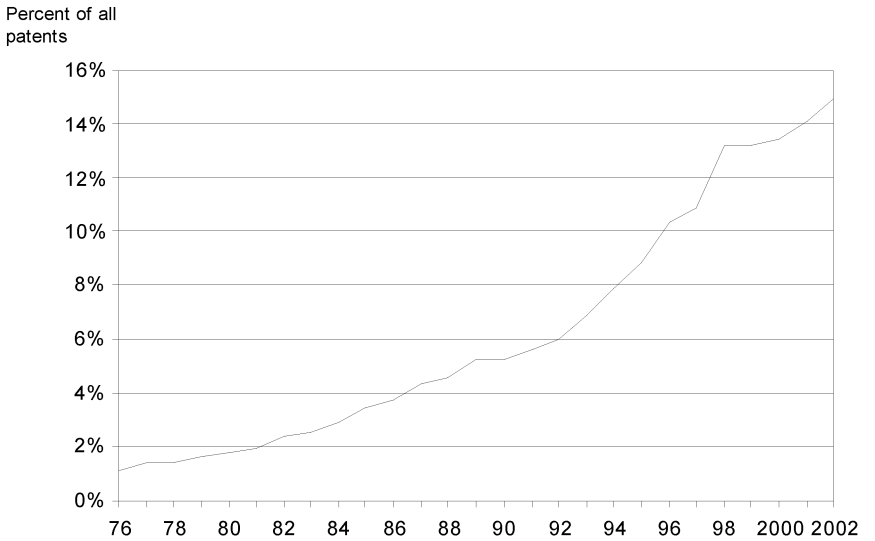


Abbildung 2: Softwarepatente in % aller Patente (Das Datum bezieht sich auf die Erteilung, nicht die Beantragung des Patents)

in dynamischen Umgebungen vorzufinden sind.

Gerade auch das Internet stellt eine hochgradig dynamische Umgebung dar, in der Innovation schrittweise erfolgt. Versuche, dem Internet neue Schutzrechte für geistiges Eigentum aufzuzwingen oder bestehende Schutzrechte zu erweitern, könnten dort unzweckmäßig sein, wo sie den Wert schöpferischer Nachahmung nicht berücksichtigen.

Wie Robert Frost in *„Mending Wall"* schrieb:

> „Bevor ich eine Mauer errichtete, würde ich fragen
> was es wäre, das ich einsperrte oder aussperrte."

Literaturverzeichnis

Bessen, J. und Hunt, R. M. (2004), 'The Software Patent Experiment', Paris. Erscheint demnächst.

Bessen, J. und Maskin, E. (2000), 'Sequential Innovation, Patents and Imitation', MIT Working Paper 00-01, http://www.researchoninnovation.org/patent.pdf [04. Feb 2005].

Das wissenschaftliche Publikationswesen auf dem Weg zu *Open Access*

SÖREN WURCH

Das wissenschaftliche Publikationswesen wird seit Jahrzehnten von einer Krise heimgesucht – der Bibliothekenkrise. Verursacht wurde diese Krise vor allem durch extreme Preissteigerungen bei Abonnements für wissenschaftliche Fachzeitschriften. In der Folge sind viele Bibliotheken nicht mehr in der Lage, ihren Lesern einen umfassenden Bestand dieser Fachzeitschriften zur Verfügung zu stellen. Die wissenschaftliche Kommunikation basiert in erster Linie auf Fachzeitschriften und wird durch deren mangelnde Verfügbarkeit stark eingeschränkt. Kommerzielle Fachverlage wollen mit ihren herkömmlichen Produktionsmethoden nicht zur Lösung dieser Krise beitragen. Daher muss ein neues Modell geschaffen werden, das auch eine strukturelle Veränderung im wissenschaftlichen Publikationswesen mit sich bringt. Ein viel versprechender Ansatz ist das Open-Access-Modell, welches zunehmend Erfolg hat. Damit wird ein freier Zugang zu wissenschaftlichen Publikationen gewährleistet, wie in diesem Artikel beschrieben wird.

1. Einleitung

Im 15. Jahrhundert waren es die Italiener, die mit dem sogenannten *Diktator*[1] über eine Methode verfügten, Texte systematisch in größerer Anzahl zu vervielfältigen (Rauner 2002). Zu Beginn des 15. Jahrhunderts waren Papier und Holzschnitttechnik in ganz Europa verbreitet, die Voraussetzungen für die massenhafte Produktion von Flugblättern mit Nachrichten der unterschiedlichsten Art mithin erfüllt (Füssel 1999, S. 10–13). Kurz darauf war es Johannes Gutenberg, der mit seiner Erfindung der beweglichen Letter (ca. 1445) die Buchproduktion revolutionierte (Fritsche 2001, S. 17).

Mit den neuen Möglichkeiten der Vervielfältigung kam auch der Gedanke auf, *Wissen* in Form von Büchern und später auch in Form von Zeitschriften als Ware

1 Der Diktator las den zu vervielfältigenden Text mehreren Schreibern vor, so dass mehrere Kopien eines Textes zeitgleich entstanden.

gegen Geld[2] zu vertreiben. Damit war die Kommerzialisierung von Wissen bereits etabliert, bevor 1665 die erste wissenschaftliche Fachzeitschrift, die „Philosophical Transactions of the Royal Society of London", herausgegeben wurde.[3]

Der Grund für die Entstehung und die Verbreitung einer wissenschaftlichen Fachzeitschrift kann in erster Linie in der immer größer werdenden Anzahl von Wissenschaftlern in vielen Ländern gesehen werden, die an der neu entstandenen wissenschaftlichen Kommunikation teilhaben wollten. Vor den ersten Fachzeitschriften griff man auf Briefe zurück – man spricht hier bereits von *kleinen Zeitschriften* –, was auf Dauer nur bei einer begrenzten Anzahl beteiligter Wissenschaftler funktionieren konnte.

2. Die Aufgaben des wissenschaftlichen Publizierens

Die Rolle des Publizierens im wissenschaftlichen Prozess wird in einer Aussage von Ebel und Bliefert (1990, SEITE) sehr schön deutlich:

> „Was immer in den Naturwissenschaften gemessen, gefunden oder theoretisiert wird – es verdient nicht, entdeckt worden zu sein, wenn es nicht anderen mitgeteilt wird."

Und nicht nur in den Naturwissenschaften ist der Austausch wissenschaftlicher Erkenntnisse unabdingbar, denn erst durch diesen Austausch entsteht die wissenschaftliche Kommunikation und diese hilft den Wissenschaftlern unterschiedlichster Disziplinen auf der ganzen Welt, neues Wissen zu produzieren. Ohne diesen Austausch kommt die Wissenschaft quasi zum Erliegen. Ebel und Bliefert (1990) sehen im schriftlichen wissenschaftlichen Bericht das eigentliche *Endprodukt der Forschung.*

Die wissenschaftliche Kommunikation erfolgt in erster Linie durch wissenschaftliche Fachzeitschriften, in denen die Wissenschaftler als Autoren die Ergebnisse ihrer Forschung den Lesern, in der Regel anderen Wissenschaftlern, mitteilen.[4] Die Qualität der wissenschaftlichen Artikel wird dabei durch das sogenannte *peer review*[5] sichergestellt.

2 Die ersten *fliegenden Blätter* zur Informationsversorgung und Wissensverbreitung gab es bereits im 15. Jahrhundert (vgl. Füssel 1999, S. 13). Nach 1600 wurden diese *fliegenden Blätter* teilweise in Zeitschriften umgewandelt und waren auf *Subskribenten* angewiesen (Burke 2001, S. 196).

3 Die Gründer bezeichneten sich vor der Etablierung der Zeitschrift und den Anfängen der wissenschaftlichen Kommunikation (ca. 1640) noch als *invisible college* und haben sich mit der Herausgabe der ersten Fachzeitschrift in die *Royal Society* umbenannt (Peek und Newby 1996, S. 5).

4 Wie bereits im vorherigen Kapitel beschrieben, treibt die wissenschaftliche Kommunikation die eigentliche Produktion von Wissen voran. Die Schlagwörter sind hier *impact* – sichtbar gemacht durch den eingeführten *journal impact factor* des *Science Citation Index´* (vgl. Wurch 2005) – und *visibility* (Sichtbarkeit), die Instrumente der Wissenschaftler umschreiben (Zimmel 2002, S. 12), um mit der Veröffentlichung unter anderem ihre *Entdeckungspriorität* anzumelden (Meier 2002, S. 20).

5 Das *peer review* ist eine Form der Qualitätskontrolle, bei der die *peers* (Kollegen des Autors) einen wissenschaftlichen Text, der in einer Fachzeitschrift veröffentlicht werden soll, begutachten und gegebenenfalls Änderungswünsche unterbreiten. Eine ausführlichere Beschreibung der Qualitätskontrolle mit Erläuterung der unterschiedlichen Formen im traditionellen und elektronischen Publizieren sowie deren Mischformen gibt Nentwich (2003, S. 367).

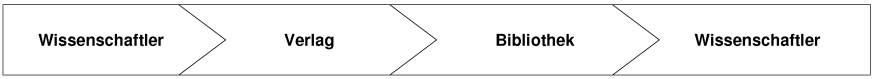


Abbildung 1: Wertschöpfungskette wissenschaftlicher Informationen (vereinfachte Darstellung; Quelle: Wurch 2005)

Das Publikationswesen im wissenschaftlichen Bereich ist seit über 300 Jahren, wie in der Wertschöpfungskette in Abbildung 1 dargestellt, aufgebaut.

Interessant ist vor allem der Fachzeitschriftenmarkt, der sich aus *scientific, technical and medical* (STM) Zeitschriften zusammensetzt, da besonders in diesem Markt die Bibliotheken mit jährlichen Preissteigerungen von 30 % und mehr für Abonnements von wissenschaftlichen Fachzeitschriften zu kämpfen haben.[6] Daher spricht man auch heute von der Bibliotheken- oder auch Zeitschriftenkrise.[7] Die essentielle Aufgabe der Bibliotheken besteht in der Bereitstellung und Archivierung von Wissen, und das schließt auch (teure) wissenschaftliche Fachzeitschriften mit ein (vgl. Neißler 2001). Sind Bibliotheken dazu nicht mehr in der Lage, wie es bereits seit Jahren an einem immer kleiner werdenden Bestand an wissenschaftlichen Fachzeitschriften zu erkennen ist, führt dies unausweichlich zu einer Krise in der Informationsversorgung. Nach Meier (2002, S. 70–73) sind die Bibliotheken für Wissenschaftler die wichtigste Quelle für den Zugang zu wissenschaftlichen Fachzeitschriften.[8] Die Krise betrifft die ganze Wissenschaft und wird sich in den kommenden Jahren weiter ausdehnen, wenn nicht neue Ansätze im Publikationswesen[9] geschaffen werden. So ist es auch nicht verwunderlich, dass die Bibliothekenkrise in all ihren Formen als *strukturelles Problem* angesehen wird, welches mit den herkömmlichen Publikationsmethoden *nicht* lösbar ist (Obst 2003).

3. Die Bibliothekenkrise in Zahlen

Nach einer Studie von BNP Paribas (2003) beträgt der jährliche Umsatz des Fachinformationsmarktes im STM-Zeitschriften-Bereich allein in den USA ca. 8 Milliarden USD, in Europa ist dieser Wert noch größer (Obst 2003). Man kann davon aber ausgehen, dass diese Zahl schon jetzt überholt ist, da die Menge der produzierten wissenschaftlichen Artikel jährlich um ca. 13 % gegenüber dem Vorjahr steigt. Nach Marek (2002) werden derzeit etwa 6 Millionen wissenschaftliche Artikel pro Jahr produziert (dies entspricht ca. 17 000 Artikel pro Tag), von denen 2,5 Millionen in einer

6 Eine genaue Betrachtung der Produktions- und Wertschöpfungskette wissenschaftlicher Informationen für Fachzeitschriften mit der Nennung der Aufgaben der Marktteilnehmer wird in (Wurch 2005) gegeben.

7 Vgl. Guédon (2001) und Nentwich (2003, S. 409).

8 Nach der Studie von BNP Paribas (2003) halten die Bibliotheken etwa in den USA 85 % der gesamten verfügbaren Menge an wissenschaftlichen Fachzeitschriften vorrätig.

9 Ein solcher Ansatz ist zum Beispiel das elektronische Publizieren von Open-Access-Zeitschriften (vergleiche Abschnitt 5. und 6.).

	1994	2004	Veränderung
Advances in Applied Mathematics	172 $	901 $	397,7 %
Journal of Lie Theory	62 $	227 $	266,1 %
Studia Logica	189 $	929 $	391,5 %
Journal of Inverse and Ill-Posed Problems	311 $	1.155 $	271,4 %
Random Structures and Algorithms	195 $	960 $	392,3 %

Tabelle 1: Preisentwicklung ausgesuchter Fachzeitschriften aus dem mathematischen Bereich zwischen 1994 bis 2004 (Quelle: American Mathematical Society 2004)

der 24 000 existierenden Fachzeitschriften – ca. 1 000 davon sind Open-Access-Zeitschriften – veröffentlicht werden (Neue Zürcher Zeitung 2004).

Die größten Preissteigerungen waren vor allem bei den sogenannten *need-to–know products*, also den als Pflichtlektüre eingestuften Publikationen der großen Fachverlage, zu verzeichnen. Allen voran kann man hier Fachzeitschriften aus dem *Reed-Elsevier*-Verlag nennen, der mit einem Marktanteil von 28,2 % am STM-Zeitschriften-Markt (vgl. House of Commons Science and Technology Committee 2004, S. 13, Abb. 2) und mit 2 115 vertriebenen Fachzeitschriften[10] eine dominante Position inne hat.[11]

So stiegen zum Beispiel die Kosten für ein Abonnement der Zeitschrift *Gene* des genannten Verlages für Institutionen außerhalb Europas/Japans von 3 924 USD im Jahre 1995 auf 9 552 USD in 2005[12] – eine Preissteigerung von 143,4 % in nur zehn Jahren. *Reed-Elsevier* ist jedoch bei weitem nicht der einzige Fachverlag, der im höchsten Maße profitorientiert arbeitet.[13]

In der Tabelle 1 sind fünf wissenschaftliche Fachzeitschriften aus dem mathematischen Bereich und deren Preisentwicklung zwischen 1994 und 2004 dargestellt. Wie man deutlich sehen kann, lagen Preissteigerungen von fast 400 % im Bereich des Möglichen, wobei nicht nur Fachzeitschriften aus dem Hause *Reed-Elsevier* betroffen waren, sondern auch von Verlagen wie *Heldermann*, *Springer*,[14] *VSP International*

10 Stand Oktober 2004; http://www.elsevier.com/wps/find/journal_browse.cws_home [28. Okt 2004].

11 Bei wissenschaftlichen Fachzeitschriften, die von Fachgesellschaften herausgegeben wurden, waren ebenfalls Preiserhöhungen in den letzten Jahren festzustellen, doch betrugen diese oft weniger als 5 % pro Jahr und sind mit der Inflationsrate oder erhöhten Produktions- und Materialkosten leicht zu erklären (vgl. House of Commons Science and Technology Committee 2004, S. 42).

12 Angaben für 1995 stammen aus Andermann und Degkwitz (2003) und für 2005 aus Wurch (2005, Kapitel 2).

13 Die Fachverlage, die den größten Teil Ihres Gesamtumsatzes mit Fachzeitschriften aus dem STM–Bereich erwirtschaften, erzielen überdurchschnittlich hohe Profite (im Vergleich zur restlichen Publikationsindustrie). Gewinnspannen von 30 % und mehr sind nicht ungewöhnlich. House of Commons Science and Technology Committee (2004) und BNP Paribas (2003).

14 Im September 2003 wurde *BertelsmannSpringer* von den europäischen Finanzinvestoren *Candover* und *Cinven* übernommen. Bereits im Januar des Jahres haben dieselben Investoren den Verlag *Kluwer Academic Publishers* gekauft. Beide Verlage wurden zusammengeführt und der neu entstandene Verlag *Springer Science+Business Media* ist hinsichtlich der Größe und Marktanteile im STM-Bereich auf

Science Publishers und *John Wiley*. Die Verschmelzung von kleineren Verlagen zu immer größeren trägt ihren Teil zur Kostenexplosion bei wissenschaftlichen Fachzeitschriften bei (McCabe 1998).

Die enormen Preissteigerungen bei den wissenschaftlichen Fachzeitschriften und die dadurch verursachten Abbestellungen von Zeitschriften durch die Abonnenten, allen voran die Bibliotheken, haben zu einem *Teufelskreis* geführt: Verlieren die Zeitschriften an Auflage, verringern sich auch die Grössenvorteile in der Produktion, was höhere Herstellungskosten zur Folge hat, die auf die Abonnementpreise umgelegt werden, was im Gegenzug zu neuen Abbestellungen führt, usw.[15] Die Fachverlage versuchen die Einnahmeverluste durch neue Vertriebsformen, wie zum Beispiel den Vertrieb von Zeitschriften in *Paketen*, den so verlorenen Profit wieder zu erwirtschaften. Dabei bieten die Verlage mehrere Zeitschriften gebündelt zu einem Gesamtpreis an, der unter den summierten Einzelpreisen der im Paket enthaltenen Zeitschriften liegt. Das Problem hierbei ist jedoch, dass die Verlage mit dieser Methode häufig versuchen, auch solche Zeitschriften zu verkaufen, die nicht zu den *need-to-know products* gehören und sich einzeln nur schwer absetzen lassen. Zudem wird von einigen Verlagen im Paket zugleich eine elektronische Version der Zeitschrift verkauft, auch wenn die eigentlich nicht gewünscht ist, wodurch sich der Preis regelmäßig erhöht und die Abonnenten gezwungen werden, zwei Versionen ein und derselben Zeitschrift abzunehmen, ohne dass daraus für sie ein wesentlicher Mehrwert resultiert.

Das Abbestellen von Fachzeitschriften ist einer der Hauptgründe, der zu weiteren Preiserhöhungen führt. Weitere Gründe sind in den Versuchen der Fachverlage zu sehen, möglichst hohe Profite zu erwirtschaften. Beispielsweise kostet die elektronische Ausgabe der Zeitschrift „New England Journal of Medicine" das 12- bis 36-fache der herkömmlichen Druckversion (vgl. Obst 2003), obwohl die Produktionskosten um bis zu 30 % gegenüber der Papierversion gesenkt werden konnten.[16]

Auch der für die Karriere so wichtige *Science Citation Index* ist durch seine Unterscheidung der Wichtigkeit von Fachzeitschriften für die Preiserhöhungen mit verantwortlich (Marx et al. 1998). Ein weiterer Einflussfaktor sind die steigenden Produktionskosten für Fachzeitschriften bei den Verlagen.[17]

Die Lage für das wissenschaftliche Publizieren ist jedoch nicht so aussichtslos, wie es in Anbetracht des Gesagten erscheint. In den folgenden Abschnitten wird gezeigt, welche erfolgreichen neuen Publikationsmodelle sich mittlerweile entwickelt haben.

Platz 2 hinter *Reed-Elsevier* vorgerückt.

15 Vgl. Wurch (2005).

16 Vgl. Odlyzko (1998) und Lacy und Anderson (2004).

17 Nach Aussagen der Verlage deckten die Einnahmen gerade einmal die Produktionskosten ab und eine jährliche Preissteigerung sei unumgänglich (vgl. House of Commons Science and Technology Committee 2004, S. 38), da die Fachverlage selbst enorm steigende Kosten für beispielsweise das *peer review* und das benötigte Produktionsmaterial (inklusive Anschaffungskosten im technischen Bereich für die Bereitstellung der Onlineausgaben der Fachzeitschriften) hätten. Doch zeigen direkte Vergleiche der Kostenstruktur für die Produktion herkömmlicher Fachzeitschriften und solcher, die nach dem Open-Access-Modell produziert werden, dass die angegebenen Produktionskosten für herkömmliche Fachzeitschriften als übertrieben gelten müssen (House of Commons Science and Technology Committee 2004, S. 40).

4. Auswege aus der Krise

Zimmel (2002) sieht drei verschiedene Ansätze zur Lösung der Bibliothekenkrise:

Konsortien Als erster Lösungsansatz ist die Bildung von Konsortien zu nennen. In den USA hat beispielsweise die *Association of Research Libraries* (ARL)[18] einige kleinere Erfolge verzeichnen können. Jedoch gibt Zimmel (2002) zu bedenken, dass diese Konsortien im Grunde eher zu einer Verlängerung der Krise als zu ihrer Lösung beitragen, führt diese Variante doch dazu, dass man den Verlagen die angestrebten Gewinne finanziert und damit den vorgängig erwähnten Teufelskreis am Leben erhält.

Etaterhöhungen Als weitere Lösungsmöglichkeit nennt Zimmel (2002, S. 9) die Erhöhung der Bibliotheksetats oder die Option der Bibliotheken, den Bestand an wissenschaftlichen Fachzeitschriften nicht jährlich an die neu herausgegebenen Fachzeitschriften anzupassen. Diese zweite Variante wird von den Bibliotheken auf Grund ihrer verzweifelten Etatsituation bevorzugt. Die Bibliotheken versuchen schon längst nicht mehr, einen (nahezu) vollständigen Bestand an wissenschaftlichen Fachzeitschriften zu besitzen (Meier 2002, S. 30).

Elektronisches Publizieren Wirklichen Erfolg verspricht jedoch nur das elektronische Publizieren, welches zusammen mit dem Open-Access-Publikationsmodell auch eine *strukturelle Veränderung* herbeiführen kann.

4.1. Elektronisches Publizieren

Beim wissenschaftlichen elektronischen Publizieren muss man zwei Hauptrichtungen unterscheiden,[19] welche beide zur Lösung der Krise beitragen. Zum einen ist das Publizieren in Open-Access-Fachzeitschriften (alternative Fachzeitschriften) zu nennen, zum anderen die elektronische *Selbstarchivierung*[20], die ebenfalls dem Open-Access–Modell folgt.[21] Die *Selbstarchivierung* wird dabei als Mittel der Vorabveröffentlichung (sog. *preprints*) und für bereits in Fachzeitschriften veröffentlichte und begutachtete Artikel genutzt.[22]

18 Siehe http://www.arl.org [28. Okt 2004].

19 Weitere Formen und Initiativen werden von Johnson (2004), Zimmel (2002), Andermann und Degkwitz (2003) und Meier (2002) erläutert.

20 Vgl. Harnad (2003).

21 Vor allem von Steven Harnard erfolgreich propagiert (vgl. Neue Zürcher Zeitung 2004).

22 Die Fachverlage versuchen zumeist, eine zusätzliche Veröffentlichung eines bereits in einer Fachzeitschrift veröffentlichten Artikels zu unterbinden. Bei der herkömmlichen wissenschaftlichen Publikation gehen Fachverlage häufig nach der sogenannten *Inglefingerrule* vor (vgl. Zimmel 2002, S. 40). Diese Regel bezeichnet die Vorgehensweise, nur Artikel zur Veröffentlichung anzunehmen, welche zuvor noch nicht öffentlich zugänglich gemacht worden sind. Hinzu kommt, dass die Verlage Buy-out-Verträge mit den Autoren abschließen (vgl. Zimmel 2002, S. 86), in denen der Autor alle Verwertungsrechte an den Verlag abtritt. Auf diese Art sichern sich die Verlage die Exklusivität und besitzen folglich ein *Alleinstellungsmerkmal* hinsichtlich der Verwertung für einen bestimmten wissenschaftlichen Artikel.

Preprint-Archive, allen voran das fast schon legendäre *ArXiV*[23], haben sich ihren Platz in der Kette der wissenschaftlichen Informationsversorgung erkämpft und sind heute nicht mehr aus dieser wegzudenken.[24]

Die technische Grundlage für die Verbreitung elektronischer Publikationen ist mit dem Internet gegeben. Durch neue Publikationsmodelle mit *Open Access* wird zudem sichergestellt, dass die Artikel praktisch kostenlos für jeden Menschen diskriminierungsfrei zugänglich sind. Damit wird unter anderem die wissenschaftliche Informationsversorgung für Länder der dritten Welt gewährleistet. Voraussetzung für den Zugang ist lediglich ein (kostengünstiger) Zugang zum Internet.[25]

5. Die Zukunft gehört *Open Access*

Am 14. Februar 2002 wurde die *Budapest Open Access Initiative* (BOAI) offiziell gestartet (vgl. Fowler 2004), welche zuvor auf Grund der beschriebenen Probleme in der Wissensversorgung von Wissenschaftlern auf einem Treffen des *Open Society Institute* (OSI) in Budapest (Dezember 2001) beschlossen worden war (vgl. Budapest Open Access Initiative 2004*a*).

Bei der *BOAI* handelt es sich um eine Übereinkunft mit dem Ziel, Initiativen und Institute im Bereich des wissenschaftlichen Open-Access-Publizierens zu koordinieren. Das Open-Access-Modell im Sinne der BOAI beinhaltet den *kostenfreien* Zugang zu wissenschaftlichen Informationen und soll einen schnellen und vor allem dauerhaften Erfolg im Kampf gegen die beschriebenen Probleme beim Zugriff auf wissenschaftliche Informationen gewährleisten (vgl. Budapest Open Access Initiative 2004*b*). Mit wissenschaftlichen Publikationen soll nach Aussage der Budapest Open Access Initiative (2004*d*) somit Folgendes möglich sein:

> „[. . .] lesen, herunterladen, kopieren, verteilen, drucken, in ihnen suchen, auf sie verweisen und sie auch sonst auf jede denkbare legale Weise benutzen können, ohne finanzielle, gesetzliche oder technische Barrieren jenseits von denen, die mit dem Internet-Zugang selbst verbunden sind."

Im Unterschied zu den herkömmlichen Ansätzen im Publikationsbereich darf der Autor den an den Verlag zur Veröffentlichung übergebenen Artikel auch weiterhin

23 Dieses wurde 1991 von dem Physiker Paul Ginsparg gegründet. Bis zum 30. Oktober 2004 wurden bereits 295 593 wissenschaftliche Artikel mit diesem System veröffentlicht (http://arxiv.org/show_monthly_submissions [30. Okt 2004]).

24 Der direkte Vergleich zwischen herkömmlichen (Papier-) Fachzeitschriften und elektronisch publizierten Fachzeitschriften zeigt, dass die elektronisch publizierten Fachzeitschriften erhebliche Vorteile mit sich bringen. Zum Beispiel werden Artikel daraus auf Grund ihrer besseren *Sichtbarkeit* im Internet ca. 2,5 mal häufiger zitiert als solche aus herkömmlichen, gedruckten Fachzeitschriften (vgl. Lawrence 2001).

25 Open-Access-Fachzeitschriften müssen nicht zwingend über das Internet vertrieben werden. Ein Ausdruck von Fachzeitschriften (oder einzelnen Artikeln) mit einem handelsüblichen Drucker ist problemlos möglich, da automatisch druckreife Dokumente erstellt werden können. Dies garantiert eine kostengünstige Versorgung mit Wissen in jenen Teilen der Welt, in denen kein oder kein bezahlbarer Internetzugang vorhanden ist (Wurch 2005).

selbst veröffentlichen. Der Verlag seinerseits stellt sicher, dass der veröffentlichte Artikel auch wirklich kostenfrei für alle Interessierten zugänglich ist. Hinsichtlich der Finanzierung unterstützt die OSI – unter anderem durch finanzielle Mittel des Gründers George Soros – Projekte, welche zu der Open-Access-Bewegung beitragen. Dies können zum Beispiel Open-Access-Fachzeitschriften sein, aber auch Finanzierungsmodelle und neue Lizenzen, die das Publizieren von wissenschaftlichen Artikeln unter dem Open-Acccess-Modell ermöglichen (vgl. Budapest Open Access Initiative 2004*e*).

Eines der wichtigsten Ereignisse für die Open-Access-Bewegung in Deutschland war die von der *Max-Planck-Gesellschaft* veranstaltete Konferenz im Oktober 2003 (vgl. Sietmann 2003), auf der die *Berliner Erklärung* beschlossen wurde, die zur Förderung des Open-Access-Publizierens aufruft (vgl. Max-Planck-Gesellschaft 2003).

In die Praxis wird das Open-Access-Modell durch alternative Fachzeitschriften und Selbstarchivierung umgesetzt.[26] Erfolgreiche Beispiele für die Umsetzung der Open-Access-Ansätze existieren bereits seit einigen Jahren, darunter mit dem Fachverlag *BioMed Central* sogar ein kommerzieller Verlag, der im folgenden Kapitel kurz vorgestellt werden soll.

6. *Open Access* kommerziell: *BioMed Central*

Der kommerzielle Fachverlag *BioMed Central* wurde 1997 gegründet und hat sich auf die Bereiche Biologie und Medizin spezialisiert. Der Verlag verfügt derzeit über ein Portfolio von ca. 120 Fachzeitschriften,[27] die alle frei nach dem Open-Access-Modell im Internet verfügbar sind.

Die Finanzierung erfolgt bei BioMed Central im Wesentlichen auf drei unterschiedlichen Wegen:

Article processing charges Es werden von den Autoren Gebühren erhoben, die pro veröffentlichtem Artikel an den Verlag zu zahlen sind. In 98 % der Fälle beträgt diese Gebühr 525 USD. Für die restlichen 2 % der Zeitschriften sind höhere Autorengebühren notwendig, da diese einen wesentlich größeren Aufwand bei der Bearbeitung von Artikeln erfordern.[28] In Ausnahmefällen, wenn sich zum Beispiel der Autor die Gebühren nachweislich nicht leisten kann, wird der Artikel auch kostenlos publiziert.

Institutional membership Eine weitere Möglichkeit ist die Mitgliedschaft von Institutionen, insbesondere Universitäten, wodurch die der Institution zugehörigen Wissenschaftler uneingeschränkt kostenlos in den Zeitschriften publizieren

26 Weitere Möglichkeiten der Veröffentlichung von wissenschaftlichen Artikeln auf Basis des Open–Access-Modells erläutert Johnson (2004).

27 Bis zum 31. Oktober 2004 wurden bereits 6 552 Artikel in den Zeitschriften von *BioMed Central* veröffentlicht (http://www.biomedcentral.com/info/about/datamining/ [31. Okt 2004]).

28 Vgl. http://www.biomedcentral.com/info/about/faq?name=revenue [28. Okt 2004].

können. Ein Beispiel: Die Kosten betragen für eine normal große Universität in Deutschland ca. 4 000 € pro Jahr (vgl. Busch 2003).[29]

Subskriptionen zu weiteren Dienstleistungen Diese nicht kostenfreien Dienstleistungen von *BioMed Central* sind vor allem in der Anfangsphase nötig gewesen, da – nach Aussagen von Busch (2003) – eine Startinvestition für den Aufbau von Open–Access-Zeitschriften unumgänglich ist.[30]

Die Qualitätskontrolle wird trotz der relativ geringen Einnahmen nicht vernachlässigt. *BioMed Central* bietet genauso wie andere Fachverlage, die nach dem klassischen Modell publizieren, das *peer review* an, doch kann sich der Autor bei *BioMed Central* zusätzlich zwischen dem traditionellen „doppeltblinden Verfahren" und dem „transparenten offenen Begutachtungsverfahren" entscheiden (vgl. Andermann und Degkwitz 2003).

Der Verlag hat auf der Basis des beschriebenen Finanzierungsmodells bereits Ende 2003 einen kleinen Gewinn erzielen können (vgl. Busch 2003). Dies ist ein zusätzliches Argument für das Open-Access-Publizieren und widerlegt die Aussagen von Fachverlagen, dass das Open-Access-Modell nicht finanzierbar wäre.[31]

Ein starkes Argument für das Open-Access-Modell ist auch die verbesserte *Sichtbarkeit* der wissenschaftlichen Artikel. So werden die Artikel von *BioMed Central* etwa 6–10 mal häufiger wahrgenommen als herkömmliche, nicht kostenfrei zugängliche Artikel.[32] Die erhöhte *Sichtbarkeit* schlägt sich auch im *Journal Impact Factor* nieder: Einige der Zeitschriften von *BioMed Central* haben bereits nach drei bis vier Jahren einen Wert von 5 erreichen können,[33] wofür klassische Fachzeitschriften Jahrzehnte gebraucht haben (vgl. Busch 2003).[34]

Der Verlag *BioMed Central* ist nur ein Beispiel von vielen für die erfolgreiche Umsetzung des Open-Access-Modells. Auch in Deutschland gibt es erfolgreiche Open–Access-Publikationen. Beispielhaft seien das *New Journal of Physics*[35], *Documenta*

29 Derzeit sind ca. 400 Institutionen weltweit Mitglied bei *BioMed Central*. Der wohl spektakulärste Neuzugang ist durch die Entscheidung Dänemarks (nach Norwegen, Finnland und Großbritannien), *BioMed Central* beizutreten (http://www.biomedcentral.com/info/about/pr-releases?pr=20041112 [17. Nov 2004]), zu verzeichnen.

30 Die *Public Library of Science* (PLoS) etwa, um ein weiteres Beispiel zu nennen, wurde unter anderem von der *Gordon and Betty Moore Foundation* mit ca. 10 Millionen USD für den Aufbau von Open-Access-Fachzeitschriften unterstützt.

31 Ergänzend sei an dieser Stelle die Berechnung in BNP Paribas (2003) genannt, in welcher von einer Autorengebühr von 1 000 USD ausgegangen wird, um Open-Access-Fachzeitschriften finanzieren zu können. Selbst dann kann die globale Forschungscommunity noch etwa 40 % gegenüber dem Erwerb von herkömmlich publizierten Fachzeitschriften einsparen.

32 Vgl. BNP Paribas (2003).

33 Laut http://www.biomedcentral.com/info/about/faq?name=impactfactor [31. Okt 2004].

34 Busch (2003) geht davon aus, dass man in weiteren drei bis vier Jahren die *need-to-know products*, die nicht auf Basis des Open-Access-Modells veröffentlicht werden, hinsichtlich des *Journal Impact Factors* überholen wird.

35 http://www.iop.org/EJ/njp [31. Okt 2004]

Mathematica[36] oder die Zeitschriften der Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF) (*German Medical Science*[37]) genannt.

7. Ausblick

Das Open-Access-Modell zum wissenschaftlichen Publizieren könnte eine Lösung für das Problem der Bibliothekenkrise darstellen. Wie das Beispiel von *BioMed Central* zeigt, lässt sich die Frage der Finanzierung selbst für kommerzielle Publikationen praktikabel beantworten. Allerdings existieren bisher zu wenige Open-Access-Fachzeitschriften, um den Bedarf zu decken.

Die Gründung einer neuen Open-Access-Fachzeitschrift ist derzeit noch schwierig, da entweder Spenden oder andere Einnahmequellen notwendig sind, um die initialen Kosten für die Einrichtung der Infrastruktur zu decken. In solchen Fällen sind staatlich geförderte Initiativen wie *Digital Peer Publishing NRW* zu begrüßen.[38]

Auch kostengünstige Alternativen zum Start einer neuen Open-Access-Fachzeitschrift sollten geprüft werden. Vor allem bei der Anschaffung der nötigen Hardware und der Software zum Erstellen wissenschaftlicher Fachzeitschriften gilt es, Alternativen zu prüfen. Universitätsbibliotheken könnten zum Beispiel enger mit den Rechenzentren der Universitäten zusammenarbeiten. Diese verfügen meist über ausreichende Hardware und auch qualifiziertes Personal zu deren Betreuung.

Im Softwarebereich sollte es ganz klar heißen: Open-Source-Software benutzen, welche durch einen Zusammenschluss von Universitäten und Forschungseinrichtungen in kürzester Zeit entwickelt werden kann und so die Investitionskosten für eine neue Open-Access-Fachzeitschrift drastisch reduzieren helfen könnte.

Auf die kommerziellen Verlage kann in Anbetracht der Bibliothekenkrise nicht gewartet werden. Zwar gibt es seit einigen Monaten auch dort erste leichte Bewegungen in Richtung *open access*. Doch die dort gewählten Ansätze sind, wie das Beispiel des Open-Choice-Modells vom Verlag *Springer Science+Business Media* mit geforderten Autorenkosten von 3 000 USD pro veröffentlichtem Artikel zeigt,[39] in der Praxis nicht tragbar. Weitere Zugeständnisse kommerzieller Verlage, wie das nachträgliche, kostenlose Veröffentlichen von wissenschaftlichen Artikeln im Internet (zwei, sechs und mehr Monate nach Erscheinen der dazugehörigen wissenschaftlichen Fachzeitschrift), tragen ebenfalls nur wenig zur Lösung der Krise bei.

Es bleibt also nur der Weg des Open-Access-Publizierens oder, im Sinne von Harold Varmus, *„Werdet Teil der Revolution!“*[40]

36 http://www.mathematik.uni-bielefeld.de/documenta/ [31. Okt 2004]

37 http://www.egms.de/de/journals/index.shtml [31. Okt 2004]

38 Im Rahmen von *Digital Peer Publishing NRW* fördert das Ministerium für Wissenschaft und Forschung von Nordrhein-Westfalen den Auf- und Ausbau von (derzeit acht) elektronischen Fachzeitschriften, wobei die Bereitstellung der Artikel unter Open-Access-Lizenzen erfolgt.

39 Vgl. http://www.springeronline.com/sgw/cda/frontpage/0,11855,5-109-2-116805-0,00.html [31. Okt 2004].

40 Vgl. Drösser (2003).

Literaturverzeichnis

American Mathematical Society (2004), 'Journal Price Survey (1994-2004)', http://www.ams.org/membership/journal-survey.html [28. Okt 2004].

Andermann, H. und Degkwitz, A. (2003), 'Neue Ansätze in der wissenschaftlichen Informationsversorgung - *Ein Überblick über Initiativen und Unternehmungen auf dem Gebiet des elektronischen Publizierens*', http://www.epublications.de/APII.pdf [28. Okt 2004].

BNP Paribas (2003), 'Professional Publishing'. Studie zur Marktentwicklung und -einschätzung am Beispiel einzelner Fachverlage (Reed Elsevier, Wolters Kluwer, etc.) von *BNP Paribas*.

Budapest Open Access Initiative (2004*a*), 'Budapest Open Access Initiative', http://www.soros.org/openaccess/g/index.shtml [28. Okt 2004].

Budapest Open Access Initiative (2004*b*), 'Budapest Open Access Initiative. Frequently Asked Questions', http://www.qualitative-research.net/fqs/boaifaq.htm [28. Okt 2004].

Budapest Open Access Initiative (2004*d*), 'Budapest Open Access Initiative. Möglichkeiten der Unterstützung', http://www.soros.org/openaccess/g/help.shtml [18. Nov 2004].

Budapest Open Access Initiative (2004*e*), 'Budapest Open Access Initiative. Unterstützung der Budapest Open Access Initiative durch das „Information Program" des Open Society Institutes', http://www.soros.org/openaccess/g/commitment.shtml [18. Nov 2004].

Burke, P. (2001), *Papier und Marktgeschrei. Die Geburt der Wissensgesellschaft*, Wagenbach, Berlin.

Busch, S. (2003), 'Open Access als Publikationsmodell in den Life Sciences - Grundlagen, Zahlen, Entwicklungen', http://digbib.ubka.uni-karlsruhe.de/diva/2004-6/ [28. Okt 2004]. Vortrag auf der Veranstaltung *Autorenrechte im Zeitalter des Elektronischen Publizierens* der Universitätsbibliothek Karlsruhe am 12. Dezember 2003. online: http://www.uni-bielefeld.de/ub/wp/docs/BioMedCentralOpenAccess.ppt [28. Okt 2004].

Drösser, C. (2003), 'Werdet Teil der Revolution', *Die Zeit* (26). Ausgabe vom 18. Juni.

Ebel, H. F. und Bliefert, C. (1990), *Schreiben und Publizieren in den Naturwissenschaften*, 3. Aufl., VCH, Weinheim.

Fowler, D. C. (Hrsg.) (2004), *E-Serials Collection Management. Transitions, Trends, and Technicalities*, The Haworth Press, New York, London, Oxford.

Fritsche, H. P. (2001), *Cross Media Publishing*, Galileo Press, Bonn.

Füssel, S. (1999), *Johannes Gutenberg*, Rowohlt Taschenbuch Verlag, Reinbek bei Hamburg.

Guédon, J.-C. (2001), In Oldenburg's Long Shadow: Librarians, Research Scientists, Publishers, and the Control of Scientific Publishing, *in* 'Creating the Digital Future', Association of Research Libraries, Washington, DC. 138th Membership Meeting, 23–25. Mai, Toronto, Ontario.

Harnad, S. (2003), 'For Whom the Gate Tolls? - *How and Why to Free the Refereed Research Literature Online Through Author/Institution Self-Archiving, Now*', http://www.ecs.soton.ac.uk/%7Eharnad/Tp/resolution.htm [28. Okt 2004].

House of Commons Science and Technology Committee (2004), 'Scientific publications: Free for all?', http://www.publications.parliament.uk/pa/cm200304/cmselect/cmsctech/399/399.pdf [28. Okt 2004]. Tenth Report of Session 2003-04 (Volume: 1).

Johnson, R. K. (2004), 'Open Access: Unlocking the Value of Scientific Research', http://www.arl.org/sparc/resources/OpenAccess_RKJ_preprint.pdf [28. Okt 2004]. In: *The New Challenge for Research Libraries: Collection Management and Strategic Access to Digital Resources*.

Lacy, M. und Anderson, M. (2004), 'New report reveals open access could reduce cost of scientific publishing by up to 30 per cent', http://www.wellcome.ac.uk/doc%5Fwtd002874.html [28. Okt 2004]. Der dazugehörige Report ist unter http://www.wellcome.ac.uk/assets/wtd003184.pdf [28. Okt 2004] zu finden.

Lawrence, S. (2001), 'Online or Invisible?', *Nature* **411**(6837), S. 521.

Marek, N. (2002), 'Intellektuellen', *Neue Nachricht* . http://www.ne-na.de/A556D3/NENA/NENA_NEU.nsf/04e0516138ca56f2c12569d2005ad117/afd99ea2aa138a55c1256b96004800ff!OpenDocument&Highlight=2,Intellektuellen [28. Okt 2004].

Marx, W., Schier, H. und Wanitschek, M. (1998), 'Kann man Forschungsqualität messen?', *MPG-Spiegel* **1998**(3).

Max-Planck-Gesellschaft (2003), 'Berlin Declaration on Open Access to Knowledge in the Sciences and Humanities', http://www.zim.mpg.de/openaccess-berlin/berlindeclaration.html [28. Okt 2004].

McCabe, M. J. (1998), 'The Impact of Publisher Mergers on Journal Prices: *A Preliminary Report*', http://www.arl.org/newsltr/200/mccabe.html [28. Okt 2004]. ARL Newsletter.

Meier, M. (2002), *Returning Science to the Scientists. Der Umbruch im STM-Zeitschriftenmarkt unter Einfluss des Electronic Publishing*, Peniope, München. zugl. Dissertation, Universität München.

Neißler, H. (2001), Erlebniswelt Medien - Bibliotheken, *in* 'Wissenschaftspublikation im digitalen Zeitalter: Verlage, Buchhandlungen und Bibliotheken in der Informationsgesellschaft', Harrassowitz, Wiesbaden, S. 139–144. Tagungsband zum Symposium des Börsenvereins des Deutschen Buchhandels e. V., der Bundesvereinigung Deutscher Bibliotheksverbände und der Deutschen Bibliothek am 8. und 9. Februar 2001 in Berlin. ISBN 3-447-04421-7.

Nentwich, M. (2003), *cyberscience - Research in the Age of the Internet*, Austrian Academy of Sciences, Wien.

Neue Zürcher Zeitung (2004), 'Open Access - aber wie? Debatte um freien Zugang zu wissenschaftlichen Publikationen', *Neue Zürcher Zeitung* . Ausgabe vom 20. Oktober.

Obst, O. (2003), 'Wissenschaftliche Verlage in Bedrängnis: Börse bestraft STM-Verleger', *med information* **2003**(4/6, Dezember).

Odlyzko, A. (1998), 'The economics of electronic journals', http://www.dtc.umn.edu/~odlyzko/doc/economics.journals.pdf [28. Okt 2004]. Fassung vom 23. Januar 1998.

Peek, R. P. und Newby, G. B. (1996), *Scholarly Publishing: The Electronic Frontier*, The MIT Press, Cambridge, Massachusetts, London.

Rauner, M. (2002), 'Wissenschaft im Cyberspace', *Neue Zürcher Zeitung* . Ausgabe vom 2. August, http://www.nzz.ch/2002/08/02/em/page-article88LHN.html [28. Okt 2004].

Sietmann, R. (2003), 'Offene Wissenschaft', *c't Magazin für Computer Technik* **2003**(23), S. 60.

Wurch, S. (2005), Elektronisches Publizieren im Internet – Prototypische Entwicklung eines Open Source Systems zur Erstellung „wissenschaftlicher Fachzeitschriften im Internet", Diplomarbeit, Fachgebiet Informatik und Gesellschaft, Technische Universität Berlin.

Zimmel, D. (2002), Wissenschaftliche Informationsversorgung im Umbruch, Diplomarbeit, FH Stuttgart.

„Anwender-Innovationsnetzwerke“: Hersteller entbehrlich

ERIC VON HIPPEL

Anwender-Innovationsnetzwerke ermöglichen die Entwicklung, Verbreitung und zum Teil auch Produktion von Innovationen – durch Anwender, für Anwender. Klassische Hersteller, der traditionellen Theorie zufolge Quelle von Innovationen, verlieren in solchen Netzwerken an Bedeutung, werden zum Teil gar entbehrlich. Zwei Beispiele für solche horizontal aufgebauten Anwender-Innovationsnetzwerke werden im Folgenden beschrieben: die Community der Entwickler im Bereich von freier und Open-Source-Software sowie die Community der Hochleistungswindsurfer. Es wird diskutiert, welche Voraussetzungen Anwender-Innovationsnetzwerke haben, und unter welchen Bedingungen sie prosperieren. Von Bedeutung sind die Existenz von Innovationsanreizen und Offenlegungsanreizen, wobei dem Verzicht auf Ansprüche aus geistigem Eigentum eine Schlüsselrolle zukommt. Der wesentliche Vorteil der Innovation in diesen Netzwerken ist – aus Anwendersicht – darin zu sehen, dass Innovationen entwickelt werden, die den Bedürfnissen der Anwender genau entsprechen. Dabei werden vorhandene Ressourcen effizienter genutzt – u. a. durch Vermeidung von „Agency-Kosten“ – als in der klassischen, vertikalen Organisation von Innovation, Produktion, Distribution und Nutzung.*

1. Einleitung

Softwareprojekte im Bereich der freien und Open-Source-Software sind aufregende Beispiele für Anwender-Innovationsnetzwerke (*user innovation networks*; d. Ü.), die von Anwendern und für Anwender aufgebaut und unterhalten werden – Hersteller entbehrlich.[1] Solche Netzwerke haben etliche Vorteile gegenüber den hersteller-

* Die Übersetzung des Textes und die Zusammenfassung stammen von Robert A. Gehring. Wir danken dem Autor, Eric von Hippel, für die freundliche Genehmigung zur Übersetzung und zum Abdruck des vorliegenden Aufsatzes. Das englische Original erscheint in J. Feller, B. Fitzgerald, S. Hissam und K. R. Lakhani, Hrsg. (2005): Perspectives on Free and Open Source Software, The MIT Press, Cambridge, MA.

1 In der Sprache der Innovationsforschung werden die Akteure danach beschrieben, in welcher Art und Weise sie erwarten, von Innovationen zu profitieren. Handelt es sich beispelsweise um Firmen oder

zentrierten Innovationsstrukturen aufzuweisen, die seit hunderten von Jahren den Kommerz dominieren. Anwender-Innovationsnetzwerke ermöglichen es jedem Teilnehmer, egal ob Individuum oder Unternehmen, genau das zu entwickeln, woran Bedarf besteht. Es ist in diesen Netzwerken nicht mehr notwendig, Hersteller als (häufig unzulängliche) Agenten in Anspruch zu nehmen. Hinzu kommt, dass nicht mehr jeder einzelne Anwender alles selbst entwickeln muss: Die Anwender können auf Innovationen zurückgreifen, die von anderen Anwendern entwickelt und der Anwendergemeinschaft frei zur Verfügung gestellt wurden.

2. Anwender-Innovationsnetzwerke

Anwender-Innovationsnetzwerke existierten schon lange vor Open-Source-Softwareprojekten und erstrecken sich auf wesentlich mehr Felder. Auch physisch greifbare Produkte werden innerhalb solcher Netzwerke entwickelt. Im Folgenden werden zwei Beispiele solcher Netzwerke im frühen Stadium beschrieben, eines aus dem Bereich der Software, das andere aus dem des Sports.

2.1. Der Apache-Webserver

Der Apache-Webserver, eine Zusammenstellung von freier und Open-Source-Software, wird im Internet eingesetzt, um auf Abruf Webseiten und andere Inhalte auszuliefern. Derartige Server bilden das Rückgrat des internetbasierten World Wide Webs.

Die Serversoftware aus der schließlich der Apache werden sollte, wurde ursprünglich von einem Studenten der University of Illinois im Grundstudium, Rob McCool, während seiner Arbeit am National Center for Supercomputing Applications (NCSA) entwickelt. McCool stellte den von ihm geschriebenen und von Zeit zu Zeit überarbeiteten Quellcode im Internet zum freien Download zur Verfügung, sodass andere Anwender ihn nutzen, modifizieren und weiterentwickeln konnten.

Als McCool das NCSA Mitte 1994 verließ, übernahm eine kleine Gruppe von Web-Administratoren die Aufgabe, die Serversoftware weiterzuentwickeln. Ein Kern von acht Leuten trug alle Dokumentationen und Fehlerbereinigungen zusammen und erstellte daraus eine konsolidierte Aktualisierung („patch"). Die so aktualisierte („patchy") Software entwickelte sich im Laufe der Zeit zum Apache-Webserver. Dank

Individuen, die sich von der Anwendung einer Innovation Profit versprechen, spricht man von „Innovationsanwendern". Als „Innovationshersteller" hingegen bezeichnet man Firmen oder Individuen, die sich Profit aus dem Verkauf einer Innovation im Markt versprechen (von Hippel 1988). Unter „Anwendernetzwerken" verstehen wir Knoten von Anwendern, die miteinander durch Kommunikationswege verbunden sind. Dabei kann es sich um persönliche Gespräche, elektronischen Informationsaustausch oder andere Kommunikationsformen handeln. Anwendernetzwerke können innerhalb der Grenzen einer geschlossenen Gruppe existieren, müssen das aber nicht. Anwender-Innovationsnetzwerke können ebenfalls – müssen aber nicht – die Qualitäten von Anwendergemeinschaften aufweisen, die folgendermaßen definiert sind: „[. . .] Netzwerke interpersonaler Verbindungen, die freundschaftliche Beziehungen, Unterstützung, Information, Zusammengehörigkeitsgefühl und soziale Identität bieten." (Wellman et al. 2002)

umfangreicher Rückmeldungen und Modifikationen von Anwendern konnte zum 1. Dezember 1995 der Apache 1.0 vorgestellt werden.

Vier Jahre später, dank vieler Ergänzungen und Verbesserungen durch seine Anwender, war der Apache die beliebteste Websersoftware im Internet, was sich auch in einer Vielzahl von Auszeichnungen niederschlägt. Trotz starker Konkurrenz kommerzieller Produkte von Microsoft und Netscape werden mit dem Apache mittlerweile über 60 % aller Websites betrieben.

2.2. Hochleistungswindsurfen

Die Entwicklung des Hochleistungswindsurfens wurde von Shah (2000) dokumentiert. Beim Hochleistungswindsurfen spielen akrobatische Einlagen wie Luftsprünge und Drehungen in der Luft eine wichtige Rolle. Früher bediente man sich eher traditioneller Elemente aus dem Segelsport und nutzte die Surfbretter im Grunde wie kleine, wendige Segelboote.

Die Grundlagen des Hochleistungswindsurfens wurden 1978 von einer Gruppe gleichgesinnter Anwender entwickelt. Ein Pionier dieses Sports, Larry Stanley, beschrieb gegenüber Shah die Entwicklung einer bahnbrechenden Innovation bei Technik und Ausrüstung:

> „Jürgen Honscheid kam 1978 aus Westdeutschland zum ersten Weltcup nach Hawaii und entdeckte dort das Springen. Zwar sprangen Mike Horgan und ich selbst bereits 1974 und 1975, aber für ihn war die Sache neu. Wir waren alle enthusiastisch, was das Springen anging, und versuchten ständig, einander auszustechen. Wir versuchten, immer höher und höher zu springen. Das Problem war, dass [. . .] man mitten in der Luft vom Brett flog, da man keinen festen Halt hatte. Im Ergebnis verletzte man sich an Füßen und Beinen, und hatte ein beschädigtes Brett.
>
> Dann erinnerte ich mich an den ‘Chip’, ein kleines experimentelles Brett mit Fußschlaufen, das wir gebaut hatten. Ich dachte, dass es eigentlich ziemlich dämlich sei, das nicht zum Springen zu nehmen. So begann ich, mit den Fußschlaufen zu springen, und entdeckte den gesteuerten Flug. Ich konnte damit viel schneller surfen als ich je gedacht hätte und wenn man eine Welle traf, war es, als ob ein Motorradfahrer auf eine Rampe fuhr: Man hob einfach ab. Mit einem Mal konnte man nicht nur in die Luft fliegen, sondern das Ding auch landen. Man konnte sogar im Flug die Richtung ändern!
>
> Von da an ging es eigentlich richtig los mit dem Hochleistungswindsurfen. Wir waren eine Gruppe von etwa zehn, die ständig miteinander surfen gingen. Sobald die anderen sahen, was ich jetzt machen konnte, montierten sie sich ebenfalls Fußschlaufen aller Art an ihre Bretter und wir rasten und

> sprangen durch die Wellen. Das löste so eine Art Schneeballeffekt aus."
>
> „1998 gab es mehr als eine Million Windsurfer, und ein Großteil der verkauften Surfbretter war mit Innovationen ausgestattet, die Anwender für den Hochleistungssport entwickelt hatten."

Mit der Zeit haben sich beide Anwendernetzwerke weiterentwickelt und sind komplexer geworden. Betrachtet man sie heute nur oberflächlich, so scheinen sie sich stark zu unterscheiden, wo sie sich doch fundamental ähnlich sind. Beide umfassen mittlerweile tausende von Nutzern. Mitglieder von Softwareprojekten im Bereich freier und Open-Source-Software interagieren hauptsächlich über das Internet. Dabei nutzen sie viele spezialisierte Websites, die ihrerseits von Freiwilligen zu diesem Zweck eingerichtet wurden. Teilnehmer innovativer Sportarten tauschen sich aus, indem sie zu ihren bevorzugten Sportstätten und zu Wettbewerben, die von ihnen ausgerichtet werden, reisen. Die meisten Anwender von freier und Open-Source-Software benutzen einfach den „Code" und verlassen sich auf die Freiwilligen, die neuen Code schreiben, Fehler in anderer Leute Code beseitigen, im Internet Fragen beantworten und Projekte koordinieren. In neu entstehenden Sportarten ist das ähnlich. Auch dort spielen die meisten einfach mit und verlassen sich auf diejenigen, die neue Techniken und Ausrüstungen entwickeln, die Entwicklungen anderer ausprobieren und verbessern, freiwillig beraten und Gruppenaktivitäten wie Ligen und Treffen organisieren (Franke und Shah 2003).

3. Ökonomie

Anwender-Innovationsnetzwerke: Sie „dürften nicht existieren", trotzdem gibt es sie

Hersteller und nicht Anwender werden traditionell als die Urheber von Produktinnovationen angesehen, die sie verkaufen. Dafür gibt es zwei Gründe:

- Auf der einen Seite hat es den Anschein, als wären die finanziellen Anreize dort für Hersteller größer als für Anwender, wo es um Innovation geht. Schließlich ist ein Hersteller in der Lage, Innovationen in einen ganzen Markt von Nutzern zu verkaufen. Hingegen werden einzelne Anwender-Innovatoren typischerweise als Nutznießer ihrer eigenen Entwicklungen eingestuft. Um von der Verbreitung innovativer Ideen in einem Markt profitieren zu können, wäre irgendeine Art von geistigem Eigentum und Lizenzierung unverzichtbar – wird angenommen. Beide sind kostspielig und der Ausgang einer solchen Unternehmung ist ungewiss.
- Auf der anderen Seite kann eine Innovation nur dann größere Reichweite erlangen, wenn auf die ursprüngliche Erfindung und Entwicklung folgend auch die Überführung in Produktion, Distribution und Vor-Ort-Service gelingt. Dafür benötigt man großindustrielle Prozesse, weshalb Hersteller im Vergleich zu

Anwendern prinzipiell den Kostenvorteil auf ihrer Seite haben müssten. Wie sollten Anwender auch in der Lage sein, ähnlich kosteneffektiv zu arbeiten wie Hersteller?

Nun, plausibel oder nicht, die Existenz von Anwender-Innovations- und Konsumptionsnetzwerken lässt sich nicht bestreiten. Mehr noch, sie erweisen sich von Fall zu Fall – man nehme das Beispiel des Apache, der mit Microsofts und Netscapes Software konkurriert – sogar als ausgesprochen durchsetzungsstark gegenüber von Herstellern produzierten Produkten. Solche Netzwerke sind nicht nur Realität – sie triumphieren!

„Und sie bewegt sich doch!“, soll Galilei gesagt haben, nachdem man ihn zum Widerruf seiner These, dass die Erde sich um die Sonne drehe, gezwungen hatte. In diesem Sinne: Was geht hier vor?

Bedingungen für Anwender-Innovationsnetzwerke

Wir behaupten, dass vollständig funktionierende Innovationsnetzwerke rein horizontal aufgebaut werden können. Die Akteure dieser Netzwerke sind alles innovative Anwender, oder, um es präziser zu formulieren, „Anwender/Selbsthersteller“. Natürlich können auch Unternehmen in diesen Netzwerken eine Rolle spielen – Red Hat und IBM sind bekannte Beispiele aus dem Open-Source-Umfeld, professionelle Sportligen und kommerzielle Hersteller sind Beispiele aus dem Bereich der Anwender–Sportnetzwerke. Unserer Meinung nach sind diese Nichtanwender aber im Grunde verzichtbar. Nur aus Anwendern bestehende – „horizontale“ – Innovationsnetzwerke sind durchaus in der Lage, Innovationen zu entwickeln, zu verbreiten, zu betreuen und zu konsumieren.

Derartige horizontale Anwender-Innovationsnetzwerke werden dann gedeihen, wenn (1) wenigstens einige Anwender hinreichende Innovationsanreize haben und innovativ agieren, (2) wenigstens einige Anwender hinreichende Anreize und Gelegenheiten haben, ihre Innovationen offen zu legen, und (3) die Anwender beim Verbreiten der Innovationen mit der kommerziellen Produktion und Distribution Schritt halten können. Sind nur die ersten beiden Bedingungen erfüllt, wird sich ein Muster herausbilden, wonach Anwender durch Versuch und Irrtum Innovationen entwickeln, deren Produktion und Verbreitung anschließend von kommerziellen Herstellern übernommen wird – insofern diese dafür eine ausreichende Nachfrage sehen.

Anwender-Innovationen

Anwender haben dann hinreichende Innovationsanreize, wenn der erwartete Nutzen die abzusehenden Kosten übersteigt. Offensichtlich trifft auf die Unterstützer von freien und Open-Source-Softwareprojekten zu, dass der zu realisierende Nutzen einen ausreichenden Innovationsanreiz bietet. Die Befunde von Niedner et al. (2000) belegen das, wenn die Programmierer in Open-Source-Projekten auf die Frage nach ihren möglichen Motiven den Nutzen durch „bessere Unterstützung meiner Arbeit

durch bessere Software“ als Hauptmotiv bezeichneten (mit einer durchschnittlichen Bewertung von 4,7 auf einer bis 5 reichenden Skala). Vergleichbar haben 59 % der Unterstützer von Open-Source-Projekten, die von Lakhani und Wolf (2001) befragt wurden, berichtet, dass der Gebrauch der so entstehenden Software für sie einen der drei größten Anreizfaktoren zur Innovation darstellt. Empirische Untersuchungen haben die Existenz von Anwender-Innovationsnetzwerken auf vielen anderen Gebieten nachgewiesen. Unter anderem haben Enos (1962), Knight (1963), Freeman (1968), Rosenberg (1976), von Hippel (1988), Shaw (1985) und Shah (2000) gezeigt, dass in vielen Fällen Anwender den Anstoß zu dem gegeben haben, was später kommerziell bedeutsame neue Produkte und Prozesse geworden sind.

Man hat auch festgestellt, dass Innovationen sogar relativ häufig von Nutzern entwickelt werden, die ein besonderes Interesse an einem bestimmten Produkt oder Prozess haben, und dass sich innovative Aktivitäten regelmäßig im Bereich der Pionier-Anwender konzentrieren (vgl. Tabelle 1).[2]

Die Forschung zu Innovationsanreizen und Innovationsfähigkeit stellt eine theoretische Basis für die empirischen Befunde bereit. Unter welchen Bedingungen Anwender Anreize zur Innovation haben – oder auch nicht –, ist untersucht worden (von Hippel 1988). Eine wichtige Rolle spielt der kostengünstige Zugang zu Informationen, die nur unter hohem Aufwand zu transferieren sind (sog. *haftende – „sticky“[3] – Information*), wie man herausgefunden hat (von Hippel 1994, Ogawa 1998): Informationen, die für erfolgreiche Innovationen unverzichtbar sind, wie z. B. Bedarf und situationsbezogene Informationen, werden vor Ort von den Anwendern *erzeugt*, sind aber nur mit großem Aufwand zu außenstehenden Entwicklern zu transferieren. Beispielsweise sind die Bedingungen, die zum Versagen von Software oder dem Scheitern von springenden Windsurfern führen, am Ort des Geschehens „umsonst“ zu evaluieren, wohingegen sie andernorts nur sehr kostenintensiv zu reproduzieren sind. Hinzu kommt, dass Informationen über Anwenderbedürfnisse und Anwendungskontext nicht statisch sind. Vielmehr entwickeln sie sich durch „learning by doing“ weiter, während die Anwender mit Prototypen experimentieren. Man erinnere sich an das

2 Die Pionier-Anwender (engl. *lead users*; d. Ü.) werden definiert als diejenigen Anwender, die zwei Eigenschaften in sich vereinen: (1) Pionier-Anwender erwarten beachtlichen Nutzen aus Innovationen, die ihren Bedürfnissen gerecht werden – was ihre Motivation begründet; (2) Pionier-Anwender artikulieren Monate oder Jahre früher Bedürfnisse, die einmal vorherrschend in einem Markt sein werden (von Hippel 1986). Man beachte, dass Pionier-Anwender nicht identisch mit den frühen Anwendern (engl. *early adopters*; d. Ü.) sind. Vielmehr befinden sie sich regelrecht vor der Adoptionskurve, da sie Bedürfnisse zum Ausdruck bringen, bevor ein entsprechendes kommerzielles Produkt existiert – weshalb sie oft eigene Lösungen entwickeln.

3 Die Haftung einer Informationseinheit in einem konkreten Fall wird definiert als die aufzuwendende, inkrementelle Ausgabe, um diese Informationseinheit an einen anderen Ort, zu einem angenommenen Informationssucher, zu transferieren. Ist der Aufwand dafür gering, so ist die Haftung der Information niedrig; ist der Aufwand hoch, so ist die Haftung hoch. Eine Reihe von Forschern hat behauptet und nachgewiesen, dass die zur Problemlösung benötigten Informationen gerade im Hochtechnologiebereich aus verschiedenen Gründen kostspielig zu transferieren sind (von Hippel 1994). Für den Fall, dass solche Informationen praktisch kostenlos zwischen Ursprungsort und Lösungsort zu transferieren sind, wird die Haftung der Informationen keinen nennenswerten Einfluss haben. Falls der Kostenaufwand für den Transfer jedoch hoch ist – in unseren Worten: die Information haftet –, wird das nicht ohne Einfluss auf die Ansiedlung der Problemlösungsaktivitäten bleiben.

Gebiet der Innovation	Stichprobe	Entwicklung und Produktion von Innovationen für eigene Zwecke (in %)	Waren die Innovatoren Pionier-Anwender?
Industrieprodukte			
Leiterplattensoftware (PC-CAD-Software) (a)	136 Teilnehmer einer PC-CAD-Konferenz	24,3	Ja
Rohraufhängungen (b)	74 Hersteller von Rohraufhängungen	36	keine Angabe
Bibliotheks-informations-systeme (BIS) (c)	102 australische Bibliotheken, die BIS einsetzen	26	Ja
Apache-Sicherheits-funktionen (d)	131 Apache-Anwender	19,1	Ja
Konsumprodukte			
Outdoor-Ausrüstung (e)	153 Besteller von Versandhauskatalogen für Outdoorausrüstung	9,8	Ja
Extremsport-Ausrüstung (f)	197 Top-Anwender	37,8	Ja
Mountainbike-Ausrüstung (g)	291 Top-Anwender	19,2	Ja

Tabelle 1: Anwenderinnovation ist häufig und findet hauptsächlich durch Pionier-Anwender statt. Quellen: (a) Urban und von Hippel (1988), (b) Herstatt und von Hippel (1992), (c) Morrison et al. (2000), (d) Franke und von Hippel (2003), (e) Lüthje (2004), (f) Franke und Shah (2003), (g) Lüthje et al. (2002)

Beispiel der Windsurfer, die erst dann *entdeckten*, dass sie ihre Surfbretter beim Flug in der Luft steuern konnten – und wollten –, *nachdem* sie mit den von ihnen entwickelten Fußschlaufen zu experimentieren begannen.

Auch aus ökonomischer Perspektive lässt sich verstehen, warum Innovationsaktivitäten unter den Pionier-Anwendern einer Anwenderpopulation konzentriert sind. Nimmt man an, dass Innovation eine ökonomische Tätigkeit sei, wird klar, dass diejenigen Anwender, die sich einen höheren Nutzen von der Entwicklung einer Innovation versprechen (wirtschaftlich oder privat), einen höheren Innovationsanreiz haben und demzufolge häufiger ensprechend aktiv werden. Da Pionier-Anwender in der Artikulation von Bedürfnissen dem Markt vorauseilen, sind für Hersteller weder das Risiko noch die potentielle Marktgröße abzuschätzen. Diese Unklarheiten verringern die In-

novationsanreize für die Hersteller und erhöhen im Gegenzug die Wahrscheinlichkeit, dass Pionier-Anwender selbst solche innovativen Lösungen entwickeln, die später im Markt verbreitete Bedürfnisse befriedigen können.

Offenlegungsanreize für Anwender

Fortschritt und Erfolg sind in Anwender-Innovationsnetzwerken darauf angewiesen, dass wenigstens einige Anwender ihre Innovationen kostenlos offenlegen.[4] Ohne kostenlose Offenlegung wären alle Anwender immer aufs Neue gezwungen, dieselbe Innovation zu entwickeln oder sie zu lizenzieren. Das würde in beiden Fällen erhebliche Kosten im System entstehen lassen.

Die Forschung hat gezeigt, dass Anwender in unterschiedlichen Domänen ihre Innovationen kostenlos offenlegen, sogar für Hersteller (von Hippel und Finkelstein 1979, Allen 1983, Lim 2000, Morrison et al. 2000, Franke und Shah 2003). Eine Demonstration der kostenlosen Offenlegung gibt das erwähnte Beispiel aus dem Sport: Innovative Anwender versammeln sich am Strand, inspizieren wechselseitig die Kreationen der anderen, die dann imitiert werden bzw. in eigene Entwicklungen einfließen. Das Resulat wird seinerseits kostenlos offengelegt. „Fast" kostenlose Offenlegung erfolgt auch bei freien und Open-Source-Softwareprojekten. (Die dabei verwendeten Lizenzen erlegen allerdings den Anwendern einige Einschränkungen auf, um die offengelegte Software im Rahmen einer „intellektuellen Allmende" (engl. *intellectual commons*; d. Ü.) verfügbar zu halten (O'Mahony 2003).

Für Ökonomen kommt die kostenlose Offenlegung überraschend, da sie eine zentrale Stütze der ökonomischen Theorie der Innovation erschüttert. Dem klassischen Verständnis zufolge, müssen Innovatoren ihr Wissen als Geschäftsgeheimnisse behandeln oder es mit Patenten schützen, um sich so den daraus abgeleiteten Mehrwert zu sichern. Unkompensierte Nutzung ihres Wissens stellte einen Verlust dar, den Innovatoren unbedingt würden vermeiden wollen, selbst wenn dazu Ausgaben erforderlich wären. Wie kommt es dann aber, dass wir Fälle freiwilliger, kostenloser Offenlegung beobachten können?

4 Mit „kostenlos offenlegen" meinen wir, dass ein Anwender alle existierenden und potentiellen Ansprüche aus geistigem Eigentum im Hinblick auf seine Innovation aufgibt. Stattdessen behandelt er die Innovation als öffentliches Gut und gewährt allen Interessenten uneingeschränkten Zugang dazu. Kostenlose Offenlegung durch den Besitzer einer Information wird in diesem Sinne definiert als die generelle Gewährung von Zugang zu Informationen ohne Verlangen einer unmittelbaren Bezahlung. Im Sinne dieser Definition wäre die Publikation von patentfreien Informationen in einer Zeitschrift oder auf einer öffentlich zugänglichen Website kostenlose Offenlegung.

Kostenlose Offenlegung, derart definiert, bedeutet allerdings nicht, dass die Beschaffung der offengelegten Informationen für Interessenten ohne Kosten zu erfolgen hat. Beschaffungskosten können beispielsweise in Form von Abonnementgebühren für Zeitschriften, Gebühren für den Internetzugang oder für Reisekosten bei einer Ortsbesichtigung anfallen. Gelegentlich mag es auch notwendig sein, komplementäre Informationen oder Ausstattungsgegenstände zu erwerben, um eine bestimmte Information vollumfänglich verstehen und in die Anwendung überführen zu können. Gemäß unserer Definition handelt es sich aber immer dann um kostenlose Offenlegung, wenn der ursprüngliche Informationsbesitzer nicht von derartigen Ausgaben zur Informationsbeschaffung profitiert (Harhoff et al. 2003).

Die Anwort auf diese schwierige Frage fällt differenziert aus. Da wäre zum einen zu berücksichtigen, dass im Fall von Software (und anderen öffentlichen Gütern) einige Aspekte selbst dann privat bleiben, wenn der Code kostenlos offengelegt wird. Diese Tatsache wurde in einem Modell der privat-gemeinschaftlichen Innovationsanreize beschrieben (von Hippel und von Krogh 2003). Um das an einem Beispiel zu illustrieren, betrachte man einige der privaten Vorteile von Anwendern, die ihren selbst geschriebenen Code kostenlos offenlegen. Dieser Code wurde möglicherweise geschrieben, um die privaten Bedürfnisse des Entwicklers zu befriedigen, kann aber gleichzeitig auch die Bedürfnisse von Trittbrettfahrern erfüllen (Harhoff et al. 2003). Wie gezeigt wurde (Lakhani und Wolf 2001), schätzen Entwickler den Lerneffekt und das Vergnügen beim Schreiben von Software als hohen Wert. Trittbrettfahrer, die den fertigen Code lediglich nutzen, können diese Erfahrung nicht teilen. Auch profitieren sie nicht von der Reputation, die ein innovativer Entwickler erwirbt (Lerner und Tirole 2002). Schließlich erwachsen dem ursprünglichen Entwickler auch aus der kostenlosen Offenlegung seines Codes zur Nutzung durch Trittbrettfahrer Vorteile: ein Teil von ihnen wird helfen, den Code von Fehlern zu bereinigen; der Code wird möglicherweise in eine offizielle Distribution integriert, was andere zur Wartung und Weiterentwicklung motiviert; mit zunehmender Nutzung (größerem „Marktanteil“) werden Netzwerkeffekte zum Tragen kommen, usw.

Ein zweiter, für die Erklärung der Praxis des kostenlosen Offenlegens bedeutsamer Aspekt ist darin zu sehen, dass es ein alles andere als triviales Unterfangen ist, kommerziell erfolgreiche Softwareprodukte zu entwickeln und zu vermarkten. Sollten die oben beschriebenen Nutzeffekte sich auf diesem Wege erreichen lassen, wäre es für ein profitorientiertes Unternehmen naheliegend, sich des Mittels der kostenlosen Offenlegung zu bedienen.

Und schließlich stellen wir fest, dass die mit einer Offenlegung verbundenen Kosten niedrig sein können. Gegebenfalls sind sie ohnehin unvermeidlich, sollten andere über dieselbe Information verfügen und sie offenlegen, falls man selbst das nicht tut. Bei niedrigen Offenlegungskosten für eine Innovation kann auch ein relativ geringer resultierender Nutzeffekt als ausreichende Belohnung empfunden werden. Kompetitive Verluste aus der kostenlosen Offenlegung hängen auch davon ab, wie hoch der Grad der Rivalität zwischen dem Softwareentwickler und potentiellen Trittbrettfahrern ist. Herrscht geringe oder keine Rivalität, hat der Entwickler von der kostenlosen Offenlegung keine Verluste zu befürchten. (Zum Beispiel herrscht zwischen Stadteilbibliotheken keine Rivalität: Sie bedienen unterschiedliche Bevölkerungsgruppen und kämpfen nicht um Marktanteile.) Gibt es mehrere unterschiedliche Personen oder Unternehmen, die vergleichbare Software entwickeln, wird die Entscheidung jedes Einzelnen von den Handlungen desjenigen maßgeblich beeinflusst werden, der am wenigsten zu verlieren hat. Sollten sie davon ausgehen müssen, dass jemand anderes die Information ohnehin kostenlos offenlegen wird, wo sie selbst eigentlich ein Interesse haben, diese vor ihren Konkurrenten geheim zu halten, werden sie die Information eventuell von sich aus offenlegen (Lakhani und von Hippel 2000).

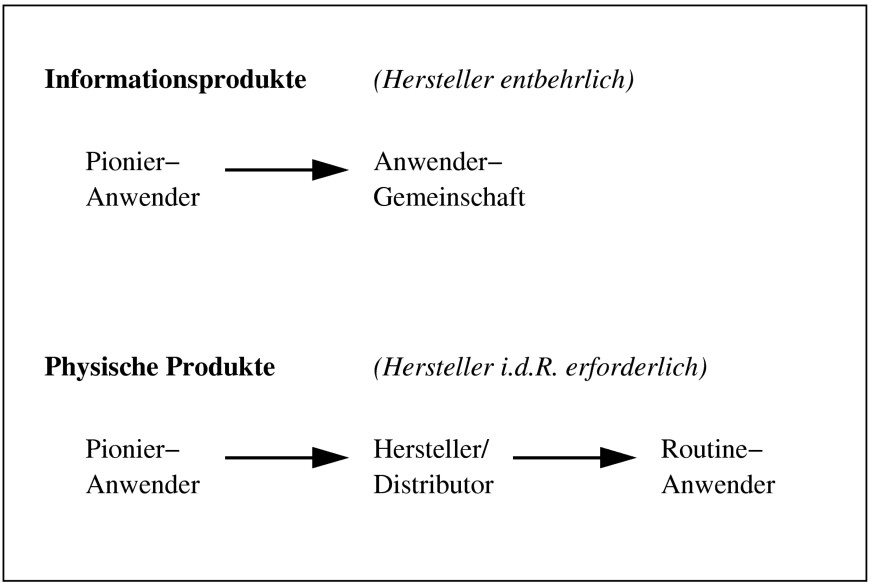


Abbildung 1: Wie Innovationen von Pionier-Anwendern Verbreitung finden

Verbreitung von Innovationen durch Anwender

„Selbsttragende" Anwender-Innovations- und Produktionsnetzwerke – „Hersteller entbehrlich" – sind nur da möglich, wo Eigenproduktion und/oder -distribution innovativer Produkte durch die Anwender im Wettbewerb mit kommerzieller Produktion und Verbreitung bestehen kann. Im Fall von freier und Open-Source-Software ist diese Möglichkeit gegeben, weil die „Produktion" und Verbreitung von Innovationen im Web praktisch kostenlos erfolgen kann, da Software ein Informationsgut und kein physisches Produkt ist (Kollock 1999). Anders stellt sich die Situation bei unserem Beispiel für Sportinnovationen dar. Dort beinhaltet die Ausrüstung physische (aber nicht „sporttechnologische") Innovationen, was sowohl industrielle Massenproduktion als auch Verbreitung erforderlich macht. Im Ergebnis können Innovationen im Bereich des Windsurfens zwar von Anwendern innerhalb von Anwender-Innovationsnetzwerken entwickelt werden, die massenhafte Integration in fertige Produkte und deren anschließende Verbreitung liegt jedoch in den Händen von Firmen (siehe Abbildung 1).

4. Die weitere Untersuchung von Anwender-Innovationsnetzwerken

Mit dem Aufkommen des Webs und der damit einhergehenden Verbreitung von freien und Open-Source-Softwareprojekten richtete sich ein erheblicher Teil akademischer Aufmerksamkeit auf Anwender-Innovationsnetzwerke im Allgemeinen und auf freie und Open-Source-Software im Besonderen. Tausende gedeihender Softwareprojekte in diesem Bereich stellen ein natürliches Experiment dar, das es Akademikern und anderen erlaubt, dieses Phänomen besser zu verstehen. Unter anderem wird jetzt untersucht, unter welchen konkreten Bedingungen freie und Open-Source-Softwareprojekte voraussichtlich Erfolg haben werden, wie sie erfolgreich zu managen sind und wann sie für Anwender attraktiv sind. Auf diesen Forschungsgebieten sind schnelle Fortschritte zu erwarten.

Für uns ist besonders faszinierend zu beobachten, dass Innovationsnetzwerke, die so gar nicht in die traditionelle Vorstellungswelt der Ökonomen passen wollen, Innovationsnetzwerke, die von Anwendern für Anwender unterhalten werden, ohne einen einzigen Hersteller in der Lage sind, komplexe Produktinnovationen hervorzubringen, zu verbreiten und zu betreuen. Daraus folgt, dass zumindest auf einigen Gebieten – und wahrscheinlich auf vielen – Anwender unabhängig von Anreizen für Unternehmen bzw. unter Vermeidung der mit der Inanspruchnahme eines Unternehmens verbundenen „Agency-Kosten“[5] nachhaltig in der Lage sind, Innovationen zu erzeugen und zu konsumieren.

Horizontale Innovationsnetzwerke – von Anwendern, für Anwender – bieten diesen in zunehmendem Maße einen praktischen und ökonomisch sinnvollen Weg, ihre Bedürfnisse zu befriedigen. Je besser wir solche Netzwerke verstehen, desto besser werden wir in der Lage sein, bestehende Netzwerke auszubauen, ihre Reichweite zu erhöhen, und ihre Vorteile zu vergrößern.

Hinweis: Aktuelle Arbeitspapiere vieler Forscher zu freier und Open-Source-Software sowie Anwender-Innovation können von den Websites http://opensource.mit.edu und http://userinnovation.mit.edu heruntergeladen werden. Diese Websites sollen alle Interessenten dabei unterstützen, sich auf dem Laufenden zu halten und eventuell unser Verständnis dieser Phänomene auszubauen.

5 Aus der Perspektive von Anwendern sind Hersteller dort deren Agenten, wo es um neue Produkte und Dienstleistungen geht. In ihren Verantwortungsbereich fällt die Aufgabe, das zu entwickeln und zu produzieren, was die Anwender zur Bedürfnisbefriedigung benötigen (schließlich stellen sie die Produkte nicht für den Eigenkonsum her). Problematisch wird diese Art der „Arbeitsteilung“, wenn die Anreize der Hersteller mit denen der Anwender nicht konform sind, was häufig genug der Fall ist. Dann zahlen die Anwender „Agency-Kosten“, wenn sie die Aufgabe der Produktgestaltung an Hersteller delegieren. Erhebliche „Agency-Kosten“ fallen beispiesweise an, wenn das vom Hersteller angebotene Produkt die Anwenderbedürfnisse nicht bestmöglich befriedigt, selbst wenn dem Hersteller diese Bedürfnisse vollständig bekannt waren. Die Hersteller bemühen sich, Entwicklungskosten auf möglichst viele Anwender zu verteilen. Im Design schlägt sich das dergestalt nieder, dass Hersteller sich an den gemeinsamen Bedürfnissen möglichst vieler Anwender statt an den individuellen Bedürfnissen eines einzelnen Anwenders orientieren.

Literaturverzeichnis

Allen, R. C. (1983), 'Collective Invention', *Journal of Economic Behavior and Organization* **4**(1), S. 1–24.

Enos, J. L. (1962), *Petroleum Progress and Profits: A History of Process Innovation*, MIT Press, Cambridge, MA.

Franke, N. und Shah, S. (2003), 'How Communities Support Innovative Activities: An Exploration of Assistance and Sharing Among End-Users', *Research Policy* **32**(1), S. 157–178.

Franke, N. und von Hippel, E. (2003), 'Satisfying heterogeneous user needs via innovation toolkits: the case of Apache security software', *Research Policy* **32**(7), S. 1199–1215.

Freeman, C. (1968), 'Chemical Process Plant: Innovation and the World Market', *National Institute Economic Review* **45**, S. 2957.

Harhoff, D., Henkel, J. und von Hippel, E. (2003), 'Profiting from voluntary information spillovers: How users benefit from freely revealing their innovations', *Research Policy* **32**(10), S. 1753–1769.

Herstatt, C. und von Hippel, E. (1992), 'From Experience: Developing New Product Concepts Via the Lead User Method: A Case Study in a „Low Tech" Field', *Journal of Product Innovation Management* **1992**(9), S. 213–221.

Knight, K. E. (1963), A Study of Technological Innovation: The Evolution of Digital Computers, PhD thesis, Carnegie Institute of Technology, Pittsburgh, PA. (Unveröffentlichte Dissertation).

Kollock, P. (1999), *The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace*, in Smith und Kollock (1999).

Lakhani, K. und Wolf, R. (2001), 'Does Free Software Mean Free Labor? Characteristics of Participants in Free and open source Communities', BCG Survey Report, http://www.osdn.com/bcg/.

Lakhani, K. und von Hippel, E. (2000), 'How Free and open source Software Works: Free User-to-User Assistance', MIT Sloan School of Management Working Paper Nr. 4117. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=290305 [04. Feb 2005].

Landau, R., Taylor, T. und Wright, G. (Hrsg.) (1996), *The Mosaic of Economic Growth*, Stanford University Press, Stanford, CA.

Lerner, J. und Tirole, J. (2002), 'Some Simple Economics of Free and open source', *Journal of Industrial Economics* **50**(2), S. 197–234.

Lim, K. (2000), 'The Many Faces of Absorbtive Capacity: Spillovers of Copper Interconnect Technology for Semiconductor Chips', MIT Sloan School of Management Working paper Nr. 4110. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=562862 [04. Feb 2005].

Lüthje, C. (2004), 'Characteristics of innovating users in a consumer goods field: An empirical study of sport-related product consumers', *Technovation* **24**(9), S. 683–695.

Lüthje, C., Herstatt, C. und von Hippel, E. (2002), 'The dominant role of local information in user innovation: The case of mountain biking', MIT Sloan School of Management Working Paper (July); http://userinnovation.mit.edu.

Morrison, P. D., Roberts, J. H. und von Hippel, E. (2000), ‘Determinants of User Innovation and Innovation Sharing in a Local Market’, *Management Science* **46**(12), S. 1513–1527.

Niedner, S., Hertel, G. und Hermann, S. (2000), ‘Motivation in Free and open source Projects’, http://www.psychologie.uni-kiel.de/linux-study/ [04. Feb 2005].

O’Mahony, S. (2003), ‘Guarding the commons: How open source contributors protect their work’, *Research Policy* **32**(7), S. 1179–1198.

Ogawa, S. (1998), ‘Does sticky information affect the locus of innovation? Evidence from the Japanese convenience-store industry’, *Research Policy* **26**(7-8), S. 777–790.

Rosenberg, N. (1976), *Perspectives on Technology*, Cambridge University Press, Cambridge, UK. (Insbes. Kap. 1, Technological Change in the Machine Tool Industry, 1840-1910).

Rosenberg, N. (1996), *Uncertainty and Technological Change*, in Landau et al. (1996), S. 334–356.

Shah, S. (2000), ‘Sources and Patterns of Innovation in a Consumer Products Field: Innovations in Sporting Equipment’, Sloan Working Paper Nr. 4105. http://opensource.mit.edu/papers/shahsportspaper.pdf [04. Feb 2005].

Shaw, B. (1985), ‘The Role of the Interaction between the User and the Manufacturer in Medical Equipment Innovation’, *R&D Management* **15**(4), S. 283–292.

Smith, M. A. und Kollock, P. (Hrsg.) (1999), *Communities in Cyberspace*, Routledge, London.

Urban, G. L. und von Hippel, E. (1988), ‘Lead User Analyses for the Development of New Industrial Products’, *Management Science* **34**(5), S. 569–582.

Wellman, B., Boase, J. und Chen, W. (2002), ‘The Networked Nature of Community On and Off the Internet’, Working paper, Centre for Urban & Community Studies, University of Toronto. aktualisierte Version erschienen in IT & Society Online Journal 1(1): 151-165, http://www.stanford.edu/group/siqss/itandsociety/v01i01/v01i01a10.pdf [04. Feb 2005].

von Hippel, E. (1986), ‘Lead Users: A Source of Novel Product Concepts’, *Management Science* **32**(7), S. 791–805.

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

von Hippel, E. (1994), ‘„Sticky Information“ and the Locus of Problem Solving: Implications for Innovation’, *Management Science* **40**(4), S. 429–439.

von Hippel, E. und Finkelstein, S. N. (1979), ‘Analysis of Innovation in Automated Clinical Chemistry Analyzers’, *Science & Public Policy* **6**(1), S. 24–37.

von Hippel, E. und von Krogh, G. (2003), ‘Open Source Software and the Private-Collective Innovation Model: Issues for Organization Science’, *Organization Science* **14**(2), S. 209–223.

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 2.0:

  Attribution (by)
  http://creativecommons.org/licenses/by/2.0/

  Attribution-NoDerivs (by-nd)
  http://creativecommons.org/licenses/by-nd/2.0/

  Attribution-NonCommercial-NoDerivs (by-nc-nd)
  http://creativecommons.org/licenses/by-nc-nd/2.0/

  Attribution-NonCommercial (by-nc)
  http://creativecommons.org/licenses/by-nc/2.0/

  Attribution-NonCommercial-ShareAlike (by-nc-sa)
  http://creativecommons.org/licenses/by-nc-sa/2.0/

  Attribution-ShareAlike (by-sa)
  http://creativecommons.org/licenses/by-sa/2.0/

  Eine Übersicht befindet sich auch unter http://creativecommons.org/licenses/.

The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . .Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Mitwirkende

Matthias Bärwolff (M. A.) ist wissenschaftlicher Mitarbeiter im Fachgebiet Informatik und Gesellschaft der Technischen Universität Berlin und hat das Kapitel „Ökonomie“ betreut.

Andreas Bauer ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Software & Systems Engineering von Prof. Broy an der Technischen Universität München. Er ist selbst langjähriger Autor freier Software und hat in verschiedenen Fachzeitschriften Artikel über deren besondere Anwendungsmerkmale und Entwicklung geschrieben. Weiterführende Informationen über Projekte und Veröffentlichungen finden sich auf seiner Homepage unter http://home.in.tum.de/baueran/.

James Bessen lehrt Recht an der Boston University School of Law. Als ein früherer Innovator schrieb er eines der ersten Programme für das Desktop-Publishing, war CEO eines Softwareunternehmens und erforscht gegenwärtig die Ökonomie der Innovation. James Bessen gibt den „Technological Innovation and Intellectual Property“-Newsletter heraus (http://www.researchoninnovation.org/tiip/index.htm).

Clemens Brandt studiert Informatik seit Oktober 2000 an der Universität Hamburg und an der Technischen Universität Berlin. Derzeitig nimmt er an einem Austauschjahr mit der McGill University in Montréal, Kanada teil. Sein Schwerpunkt im Studium ist Informatik und Gesellschaft und er hat für das Jahrbuch das Kapitel „Open Content“ betreut.

Nicole Bräuer (M. A.) arbeitet derzeit bei DaimlerChrysler im Bereich Markenkommunikation. Wir konnten sie dafür gewinnen, das Lektorat für weite Teile des Buches durchzuführen.

Horst Bräuner ist seit 1989 IT-Leiter bei der Stadt Schwäbisch Hall. Zudem ist er Geschäftsführer der tosipa-SHA GmbH. Zuvor absolvierte er die Ausbildung zum Diplom Verw. Wirt (FH). Er besuchte verschiedenste Seminare im IT-Umfeld u. a. zum *Certified Security Engineer*. Des weiteren ist Horst Bräuner als Projektleiter, Dozent und Internet-/Netzwerk-Verantwortlicher tätig.

Carsten Brunke (Dipl. Inform.) ist Gründer und Geschäftsführer der inmedias.it GmbH (http://www.inmedias.it). Nach seinem Studium an der FH Gießen-Friedberg arbeitete er zunächst in der IT-Abteilung eines großen Medienkonzerns, um sich dann 1995 als IT-Consultant selbstständig zu machen. Mit der 1998 erfolgten Gründung der inmedias.it GmbH in Hamburg hat sich sein Wirken auf Freie Software fokussiert. So ist Carsten Brunke heute im Arbeitskreis Open Source der Hamburger Initiative Hamburg@Work und im Linux-Verband e. V. (LIVE) aktiv. Mit seinem Unternehmen tritt er als Träger der Free Software Foundation Europe (http://www.fsfe.org) auf.

Patrick Danowski ist Dipl. Informatiker und seit Oktober 2004 Bibliothekreferendar an der Zentral- und Landesbibliothek Berlin. Er beschäftigt sich mit Digitalisaten und dem elektronischen Publizieren.

Thomas Ebinger (LL. M. Rechtsinformatik) arbeitet als Rechtsanwalt in Berlin insbesondere im IT-Recht und ist Partner der Kanzlei Berger Groß Höhmann. Er hat bereits mehrfach zu rechtlichen Fragen von Open-Source-Software veröffentlicht; nach Studium und Referendariat in Berlin erwarb er mit Zusatzstudium der Rechtsinformatik/IT-Law in Hannover und London (http://www.eulisp.de) seinen Master of Laws (LL. M.) 2001 mit einer Abschlussarbeit über Open-Source-Software und Softwarepatente; 2001 bis 2003 arbeitete er als Rechtsanwalt in München in einer renommierten internationalen Kanzlei im IT-Recht.

Oliver Feiler hat Abitur, ist selbständig und arbeitet in einer kleinen Firma, die nach einem Richtungswechsel jetzt Software für den Apple Macintosh entwickelt. Sämtliche Programmierkentnisse stammen aus Eigeninitiative. Sein größtes Interesse gilt aber nach wie vor der Entwicklung von Computerspielen und nebenbei der Fotografie. Von ihm stammt der in diesem Buch abgedruckte Sourcecode. Neben dem POP3-Server Snowbox hat er zahlreiche weitere nützliche Programme geschrieben und ist auch in andere Open-Source-Projekte involviert.

Robert A. Gehring studierte Elektrotechnik, Informatik und Philosophie an Technischen Hochschule Ilmenau und an der Technischen Universität Berlin. Nach dem Studium arbeitete er freiberuflich als Consultant, Dozent und Autor, bevor er an die Technischen Universität Berlin zurückkehrte. Dort arbeitete er im Fachgebiet Informatik und Gesellschaft als wissenschaftlicher Mitarbeiter mit den Forschungsschwerpunkten Open Source, IT-Sicherheit und „geistiges Eigentum". Er promoviert zu Fragen der Softwareökonomie und des Softwarerechts. Neben der Wahrnehmung der Aufgaben als Herausgeber hat Robert Gehring für dieses Jahrbuch das Kapitel „Open Innovations" betreut.

Christian F. Görlich (Studiendirektor) machte an der Universität in Münster sein Staatsexamen in den Fächern Deutsch, Philosophie, Soziologie und Erziehungswissenschaften. Er ist Leiter des Seminars Gymnasium/Gesamtschule im Studienseminar für Lehrämter an Schulen in Hamm. Nach einer zeitweiligen Abordnung an das Schulkollegium in Münster übernahm er in der Lehrerausbildung die Fachleitung für die Fächer Sozialwissenschaften und Philosophie am Studienseminar für die Sekundarstufe II in Hamm. Er ist weiterhin wissenschaftlicher Mitarbeiter (Zentrum für interdisziplinäre Forschung, Bielefeld) im Forschungsprojekt „Theoriebildung als Gruppenprozeß" (Zuständigkeitsbereich Psychoanalyse) und ist Lehrbeauftragter für Philosophie an den Universitäten in Münster und Paderborn.

Annika Held studiert seit 2004 Medienberatung an der Technischen Universität Berlin. Zuvor hat sie einen Bachelor of Arts in Philosophy and Economics an der Universität Bayreuth absolviert und ihre Abschlussarbeit über Open-Source-Software

als Herausforderung des Rechts „geistigen Eigentums" geschrieben. Für dieses Buch hat sie Marketingaufgaben übernommen.

Joachim Henkel (Prof. Dr.) ist Inhaber des Dr. Theo Schöller Stiftungslehrstuhls für Technologie- und Innovationsmanagement an der Technischen Universität München. Er studierte in Bochum und Bonn, promovierte am Graduiertenkolleg „Allokation auf Finanz- und Gütermärkten" der Universität Mannheim und habilitierte sich 2004 an der Ludwig-Maximilians-Universität München mit einer Arbeit zu Open-Source–Aktivitäten von Unternehmen. Er ist verheiratet und hat zwei Kinder.

Jens Herrmann studierte Rechtswissenschaften und Informatik in Berlin. Er beendete sein Studium an der Technischen Universität Berlin mit einer Diplomarbeit zum Thema Vorratsdatenspeicherung im Internet. Zur Zeit arbeitet er zu den Themen Datenschutz, Urheberrecht in der Informationsgesellschaft sowie Freier Software.

Maik Hetmank (Diplom Volkswirt) studierte bis 2004 an der Universität Duisburg-Essen. Die Schwerpunkte seines Studium setzte er in den Bereichen der Sozialen Sicherung und der Besteuerung. Nebenher interessiert er sich schon seit frühester Kindheit für Computer. Seit einigen Jahren befasst er sich auch intensiv mit Open–Source-Software und hat über dieses Thema seine Diplomarbeit verfasst.

Ludger Humbert (Dr. rer. nat.) studierte bis 1976 Informatik an der Universität Paderborn. Nach seinem zweiten Studium wurde er Informatiklehrer. An der Universität Siegen promovierte er 2003 in der Didaktik der Informatik mit dem Thema „Zur wissenschaftlichen Fundierung der Schulinformatik". Derzeit ist er Informatiklehrer an der Willy-Brandt-Gesamtschule in Bergkamen und Informatikdidaktiker und Lehrerbildner am Studienseminar Hamm. Er beschäftigt sich u. a. mit wissenschaftstheoretischen Fragen und Standards zur informatischen Bildung. Darüber hinaus arbeitet er im Aufgabenausschuss für den Bundeswettbewerb Informatik.

Christian Imhorst studierte bis 2004 Politische Wissenschaft und Philosophie an der Universität Hannover. Seine Magisterarbeit schrieb er über „Richard Stallman und die freie Software-Bewegung", die Anfang 2005 unter dem Titel „Die Anarchie der Hacker" als Buch erschien.

Stefan Krempl (Dr.) Jahrgang 1969, arbeitet als freier Autor in Berlin und als Dozent am Südosteuropäischen Medienzentrum in Sofia. Er publiziert regelmäßig in Magazinen und Online-Diensten wie c't, heise online und Telepolis sowie in Tages- und Wochenzeitungen von der Financial Times Deutschland über die Neue Zürcher Zeitung und die VDI nachrichten bis zur Zeit schwerpunktmäßig über politische und rechtliche Themen rund um Internet und Informationstechnik. Buchveröffentlichungen: „Medien, Internet, Krieg: Das Beispiel Kosovo." (2004), „Krieg und Internet: Ausweg aus der Propaganda?" (2004) und „Das Phänomen Berlusconi" (1996).

Christiane Kunath arbeitet seit September 1991 als Systemadministratorin bei der BStU in Berlin. Neben der Administration verschiedener Betriebssysteme (Novell

Netware, Windows 2000, SuSE Linux) gehörte in den letzten Jahren vor allem der Aufbau des Intranets und des Firewallsystems der BStU zu ihrem Aufgabenbereich. Seit Oktober 2002 war sie als stellvertretende Projektleiterin im Referat IT/TK für die konzeptionelle Erarbeitung und Durchführung von Teilschritten der Migration der Server und Clients bei der BStU verantwortlich.

Katja Luther ist Diplombiologin und Studentin der Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Informatik und Gesellschaft" und „Softwaretechnik". Zur Zeit schreibt sie an ihrer Diplomarbeit zum Thema „künstliche Immunsysteme". Sie hat für dieses Buch das Kapitel „Recht und Politik" betreut.

Bernd Lutterbeck (Prof. Dr. iur.) studierte Rechtswissenschaften und Betriebswirtschaftslehre in Kiel und Tübingen. Danach folgten wissenschaftliche Tätigkeiten an den Universitäten Regensburg (1969–1971, Fachbereich Rechtswissenschaft) und Hamburg (1974–1978, Dozent am Fachbereich Informatik) sowie 1976 die Promotion zum Dr. iur. an der Universität Regensburg. Von 1978–1984 war er Beamter beim Bundesbeauftragten für den Datenschutz in Bonn. Seit 1984 ist Bernd Lutterbeck Professor für Wirtschaftsinformatik an der Technischen Universität Berlin mit den Schwerpunkten Informatik und Gesellschaft, Datenschutz- und Informationsrecht, Verwaltungsinformatik. Seit 1995 nimmt er für die Action Jean Monnet der Europäischen Union Brüssel an der Technischen Universität Berlin eine Professur für humanwissenschaftliche Fragen der europäischen Integration wahr. Seine aktuelle Arbeitsschwerpunkte sind E-Government, Theorie und Praxis der *property rights*, Open Source und *European Governance*. Er ist einer der Herausgeber dieses Jahrbuchs.

Christian Maaß (Dipl.-Kfm.) ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Betriebswirtschaftslehre, insbesondere Organisation und Planung, an der FernUniversität Hagen.

Eric Maskin ist A.O.-Hirschman-Professor Sozialwissenschaften am Institute for Advanced Study der Universität Princeton. Vorher unterrichtete er am MIT und der Harvard-Universität. Seine Arbeitsgebiete umfassen verschiedene Bereiche der Ökonomik, einschließlich Spieltheorie, Anreiztheorie und Theorie des gemeinschaftlichen Handelns (*social choice*).

Jason Matusow ist der Direktor der Shared-Source-Initiative von Microsoft. Er ist verantwortlich für die Business Strategy und Implementierung des globalen Programmes zur Lizenzierung von Microsoft Source Code und arbeitet dazu mit Regierungen, Unternehmen und Universitäten zusammen.

Stefan Meretz machte zuerst 1989 seinen Abschluss als Diplom-Ingenieur an der Technischen Universität Berlin. Vier Jahre später absolvierte er erfolgreich das Studium der Informatik (Diplom) und Promotion (Ing.). Er ist derzeit tätig bei „Vereinte Dienstleistungsgewerkschaft" und begleitet dort das Projekt di.ver. Seine Schwerpunkte liegen insbesondere im Projekt Oekonux (http://oekonux.de), Open Theory (http://opentheory.org) und Kritische Informatik (http://kritische-informatik.de) sowie Freie Software.

Stefan Merten studierte Informatik an der Universität Kaiserslautern und machte dort 1992 sein Diplom. Derzeit arbeitet er als technischer Projektleiter bei einer Internetfirma. Neben der Teilnahme an kleineren Freie-Software-Projekten ist er beteiligt am Projekt Oekonux (http://www.oekonux.de) und Open Theory (http://www.opentheory.org).

Jörg Meyer (Dr.) studierte Physik an der Universität Hamburg und promovierte dort bis 1992. Danach begann er seinen beruflichen Werdegang als Systementwickler und Projektleiter bei der has program service GmbH. Seit 1996 arbeitet er bei der Norddeutschen Affinerie AG (http://www.na-ag.com). Nach zwei Jahren als Systemanalytiker wurde er dort Leiter u. a. der Netzwerke und Datenbankentwicklung und seit 2004 auch der gesamten IT-Infrastruktur inklusive der Telekommunikationsanwendungen.

Jan Mühlig (M. A.) ist Vorstand und Gründer der relevantive AG (relevantive.de) und u. a. Mitautor des „Linux Usability Report" von 2003. Er studierte in Regensburg, Mainz, Chicago und Lausanne und hat einen Magister in Soziologie. Sein Interesse gilt insbesondere der projektbegleitenden Usability und damit der Frage, in welchen Prozessen die Benutzerfreundlichkeit für die Open-Source-Entwicklung integriert werden kann.

Jens Mundhenke ist Dipl.-Volkswirt und Dipl.-Kaufmann, studierte Volks- und Betriebswirtschaftslehre an der Universität Passau, dem University College Dublin und der Christian-Albrechts-Universität zu Kiel. Seit 2000 ist er als Wissenschaftlicher Mitarbeiter am Institut für Weltwirtschaft an der Universität Kiel tätig und forscht dort im Bereich der Wettbewerbspolitik in der Neuen Ökonomie sowie der Informations- und Netzwerkgüter. Er promoviert zu Fragen der ökonomischen Bedeutung und der Wirkung von Open Source Software.

Andreas Neumann ist wissenschaftlicher Mitarbeiter am Zentrum für Europäische Integrationsforschung an der Rheinischen Friedrich-Wilhelms-Universität Bonn und Mitglied der Forschungsprojektgruppe „Telekommunikationsrecht". Er gehört zum Redaktionsteam von tkrecht.de, einer WWW-Seite zum deutschen und europäischen Telekommunikations- und Medienrecht, sowie zum Redaktionsteam der Medienrechtsseite artikel5.de.

Denis Petrov (Dipl. Informatiker) studierte an der Russischen Staatlichen Geisteswissenschaftlichen Universität (RGGU Moskau), arbeitet seit 2000 bei der Tembit Software GmbH als Softwareentwickler und Berater im eHealthcare Bereich.

Markus Pizka (Dr.) ist wissenschaftlicher Assistent am Lehrstuhl für Software & Systems Engineering der Technischen Universität München. Im Rahmen seiner Promotion über verteilte Betriebssysteme hat er von 1995–1999 u. a. die OSS-Produkte GNU GCC und Linux adaptiert und eigene OSS-Projekte initiiert. Seit 2001 konzentriert sich seine wissenschaftliche Arbeit auf die Wartung und Evolution großer Softwaresysteme.

Sebastian Redenz lebt in Mannheim und studiert an der dortigen Universität Volkswirtschaftslehre. In seiner Freizeit schreibt er Rezensionen für *De:Bug*. Sebastian Redenz war in früheren Module Groups ab 1998 involviert und veröffentlichte auf einigen Musik. Seit 2001 betreut er mit *Thinner* und *Autoplate* zwei der bekanntesten Netlabel Projekte (http://www.thinnerism.com).

Wolfram Riedel ist Magisterstudent im Hauptstudium an der Technischen Universität Berlin. Seine Hauptfächer sind Kommunikationswissenschaft (Grundlagen von Sprache und Musik) und Informatik (Schwerpunkt Informatik und Gesellschaft). Für dieses Buch hat er das Layout erstellt und das Kapitel „Technik" betreut.

Jan Schallaböck ist zur Zeit Rechtsreferendar am Landgericht Neuruppin. Er war im vergangenen Jahr als Mitarbeiter der Heinrich-Böll-Stiftung mit dem zivilgesellschaftlichen Monitoring des UN-Weltgipfels der Informationsgesellschaft in Genf befaßt. Zuvor war er unter anderem beteiligt an dem Projekt *Bundestux*, das sich für den Einsatz von freier Software in der Bundestagsverwaltung einsetzte, sowie an der Konferenz *Save Privacy*, die die Erosion des von Bürgerrechten zur Zeit der ersten Anti-Terror-Gesetzgebung analysierte.

Ewald Scherm (Univ.-Prof. Dr.) ist Inhaber des Lehrstuhls für Betriebswirtschaftslehre, insbesondere Organisation und Planung, an der FernUniversität Hagen.

Broder Schümann studiert Informatik (Dipl.-Ing.) an der Universität Kiel. Zunächst als Praktikant, dann als freier Mitarbeiter an GENOMatch beteiligt, arbeitet er derzeit im Rahmen seiner Diplomarbeit an einer Erweiterung zu GENOMatch.

Walter Seemayer ist der National Technology Officer der Microsoft Deutschland GmbH. In dieser Funktion ist er für die Unterstützung und Zusammenarbeit mit mit dem Public Sector zuständig. Seine Themenschwerpunkte sind Microsofts Technologie Vision und Strategie, High-Tech Start-Ups, Kosten und Nutzen von IT-Investitionen, Offene Standards, Sicherheit und Datenschutz. Er engagiert sich in verschiedenen Gremien und Lenkungsausschüssen, die sich mit Thema Innovation im IT Bereich beschäftigen wie z. B. dem Münchner Kreis, der Software Offensive Bayern und dem Beirat des Fraunhofer e-Government Zentrums.

Patrick Stewin studiert seit Oktober 2000 Informatik an der Technischen Universität Berlin. Die Schwerpunkte seines Studiums setzt er in den Fachgebieten Intelligente Netze und Management Verteilter Systeme (im Studiengebiet Betriebs- und Kommunikationssysteme) und Informatik und Gesellschaft (am Institut für Wirtschaftsinformatik). Im Rahmen seines Studiums absolviert er im Wintersemester 2004/05 ein Praktikum bei der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern und arbeitet dort am Open-Source-Software-Kompetenzzentrum mit. Er wirkte bereits am „Open Source Jahrbuch 2004" mit und hat in diesem Buch des Kapitel „Fallbeispiele" betreut.

Joachim Sturm (Dr.) ist seit April 2004 Leiter der KBSt (Referat IT 2 im IT-Stab des BMI). Aufgabe der KBSt ist primär die Beratung der obersten Bundesbehörden beim Einsatz von IuK. Sie gibt dazu Empfehlungen heraus und koordiniert wesentliche Entwicklungsvorhaben. Sie ist darüber hinaus auch für die Regierungskommunikation (IVBB) zuständig . Des weiteren zeichnet sie sich verantwortlich für Standadisierungsfragen (SAGA) und den Einsatz von OSS in der Bundesverwaltung (Migrationslitfaden). Als Leiter der KBSt ist Herr Sturm Vorsitzender des IMKA und Vertreter des Bundes im KoopA ADV.

Mark Tins (Dipl.-Kaufm.) studierte Betriebswirtschaftslehre an der Ludwig-Maximilians-Universität München. Seine Schwerpunkte waren Innovationsmanagement am Lehrstuhl von Prof. Dietmar Harhoff und Informations- und Kommunikationsmanagement am Lehrstuhl von Prof. Arnold Picot. Im Rahmen seiner Ausbildung verbrachte er jeweils einen mehrmonatigen Studienaufenthalt in Ohio, USA und an der Katholieke Universiteit in Leuven, Belgien. Im Februar 2004 schloss er sein Studium mit der empirischen Diplomarbeit zum Thema „Informelle Entwicklungskooperationen in der Embedded Software Branche" ab. Seit Anfang 2004 ist er im Produktmanagement in der Mobilfunkbranche tätig.

Siva Vaidhyanathan (http://sivacracy.net) ist der Autor von den Büchern „Copyrights and Copywrongs: The Rise of Intellectual Property and How it Threatens Creativity" (New York University Press, 2001) und „The Anarchist in the Library: How the Clash Between Freedom and Control is Hacking the Real World and Crashing the System" (Basic Books, 2004). Er hat bereits eine Vielzahl an Beiträgen für Zeitschriften wie *The Chronicle of Higher Education*, *The New York Times Magazine*, *MSNBC.COM*, *Salon.com*, *openDemocracy.net* und *The Nation* geschrieben. Nach fünfjähriger Tätigkeit als professioneller Journalist hat Vaidhyanathan an der University of Texas in Amerika Studien promoviert. Er hat bisher an der Wesleyan University und an der University of Wisconsin gelehrt und ist derzeitig Dozent fuer Kultur- und Kommunikationswissenschaften an der New York University.

Tile von Damm leitet die unabhängige Analyseorganisation Perspektiven Globaler Politik (*PerGlobal*). Er arbeitet und forscht zur Ausgestaltung der Europäischen Union, zu internationalen und europäischen Umwelt- und Nachhaltigkeitsprozessen, zur Stärkung einer euopäischen Zivilgesellschaft, zu Partizipationsprozessen sowie zur Informationsgesellschaft. Als NGO-Vertreter war er auf den UN-Gipfeln zur Nachhaltigen Entwicklung und zur Weltinformationsgesellschaft. Er studierte Politikwissenschaften in Berlin und Marburg und war 1996-1999 bei *Sony Music Germany* (http://www.perglobal.org, http://www.nachhaltiges-europa.de).

Eric von Hippel leitet die Forschungsgruppe für Technological Innovation and Entrepreneurship an der MIT Sloan School of Management. Seine Forschungsschwerpunkte sind die Quellen und Ökonomie von Innovation.

Jakob Voß studiert Informatik und Bibliothekswissenschaft an der Humboldt–Universität zu Berlin. Er ist Mitglied des Vorstands von *Wikimedia Deutschland e. V.*

Sören Wurch ist Student der Informatik an der Technischen Universität Berlin. Zur Zeit stellt er seine Diplomarbeit zum Thema „elektronisches Publizieren“ fertig.

Sebastian Ziebell studiert seit Oktober 2000 Informatik an der Technischen Universität Berlin. Die Schwerpunkte in seinem Studium sind „Softwaretechnik“ und „Informatik und Gesellschaft“. Er hat das Kapitel „Gesellschaft“ betreut.

Bastian Zimmermann ist Student an der Technischen Universität Berlin. Die Schwerpunkte in seinem Studium sind „Softwaretechnik“ und „Informatik und Gesellschaft“. Er hat das Qualitätsmanagement und die Übersetzung verschiedener Artikel im Buch übernommen.

Auch zu empfehlen – Das *Open Source Jahrbuch 2004*
Erhältlich unter http://www.opensourcejahrbuch.de