Bernd Lutterbeck
Matthias Bärwolff
Robert A. Gehring (Hrsg.)

Open Source

Jahrbuch 2007

Zwischen freier Software und Gesellschaftsmodell

Open Source Jahrbuch 2007

Zwischen freier Software und Gesellschaftsmodell

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

Open Source Jahrbuch 2007

Zwischen freier Software und Gesellschaftsmodell

Herausgegeben und bearbeitet von:

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

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

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

Daniel Auener
Student der Informatik, TU Berlin

Fabian Flägel
Student der Informatik, TU Berlin

Martin Görnitz
Student der Informatik, TU Berlin

Hans Henning
Student der Informatik, TU Berlin

Matthias Liebig
Student der Informatik, TU Berlin

Nadim Sarrouh
Student der Informatik, TU Berlin

Gerhard Schöppe
Student der Informatik, TU Berlin

Sebastian Stein
Student der Informatik, TU Berlin

Silvia Varadinova
Studentin der Informatik, TU Berlin

2007

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 2007
Zwischen freier Software und Gesellschaftsmodell
Lutterbeck, Bernd; Bärwolff, Matthias; Gehring, Robert A. (Hrsg.)

Berlin, 2007 – Lehmanns Media – LOB.de
ISBN: 978–3–86541–191–4

© für die einzelnen Beiträge bei den Autoren
© für das Gesamtwerk bei den Herausgebern
Das Werk steht elektronisch im Internet zur Verfügung:
<http://www.opensourcejahrbuch.de/>

Satz und Layout: Wolfram Riedel, Matthias Liebig
unter Verwendung des Textsatzsystems LaTeX
Einbandgestaltung: Nadim Sarrouh, Matthias Bärwolff
unter Verwendung des Textes der *GNU General Public License*
Druck und Verarbeitung: Druckhaus Berlin-Mitte

Dieses Buch wurde in der Europäischen Union
und auf chlorfrei gebleichtem Recycling-Papier gedruckt.

Für Cosima Noëmi Florette und Linus Noam, die im Laufe dieses Projektes geboren wurden.

Inhalt

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

Warum „Open Source" das Wesentliche von „Freier Software" verdeckt. . . . 1
von der Free-Software-Bewegung: Richard Stallman

Die ökonomischen Grenzen freier Software 9
von Matthias Bärwolff

Teil I Open Source fasziniert

Kapitel 1 Das Prinzip Open Source

Das Prinzip Open Source: Einleitung 15
von Daniel Auener

Offene Geheimnisse – Die Ausbildung der Open-Source-Praxis im 20. Jh. . . 17
von Gundolf S. Freyermuth

Die Deutungsoffenheit der Quelle. 59
von Alexander Knorr

Pharmaforschung mit Open-Source-Methoden. 73
von Janet Hope

Interaktive Wertschöpfung – Produktion nach Open-Source-Prinzipien 87
von Frank Piller, Ralf Reichwald und Christopher Ihl

Piazze telematiche, Video Bridges, Open Coops – der mühsame Weg zu den Globalen Dörfern 103
von Franz Nahrada

Kapitel 2 Open-Source-Ökonomie

Open-Source-Ökonomie: Einleitung. 123
von Daniel Auener

Die Ökonomie der Softwaremärkte 125
von Hal Varian und Carl Shapiro

Open Source – ein aufstrebendes ökonomisches Modell 131
von Bruce Perens

Warum ist Open-Access-Entwicklung so erfolgreich? Stigmergische Organisation und die Ökonomie der Information 165
von Francis Heylighen

Teil II Open Source funktioniert

Kapitel 3 Open Source: Die Entscheidung

Die Entscheidung: Einleitung 183
von Hans Henning

Entwicklung von Open-Source-Software: Kostenrelevante Eigenschaften einer ungewöhnlichen Organisationsform 185
von Jürgen Bitzer und Philipp J. H. Schröder

Open-Source-Software – Einsatzstrategien, Reifegrad und Wirtschaftlichkeit . 195
von Michael Vetter, Thomas Renner, Holger Kett und Sascha Rex

Einführung von Linux im öffentlichen Sektor 209
von Hal Varian und Carl Shapiro

Linux fensterlt in der Verwaltung 223
von Stefan Krempl

World Domination: Die Erfolgsgeschichte der Linux- und Open-Source-Einführung im Auswärtigen Amt 239
von Torsten Werner

Nach der Migration – business as usual? 249
von Carsten Brunke und Jörg Meyer

Kapitel 4 Faszination Technik

Faszination Technik: Einleitung 261
von Nadim Sarrouh

OpenVZ und Xen 263
von Dennis Zimmer und Andreas Niederländer

Prototyp eines Open-Source-basierten Thin-Clients 273
von Nico Gulden

Einbeziehung von Usability-Experten in Open-Source-Community-Projekte: Erfahrungen aus dem OpenUsability-Projekt 289
von Ellen Reitmayr

Turaya – Die offene Trusted-Computing-Sicherheitsplattform 299
von Norbert Pohlmann und Markus Linnemann

NetBSD – Das portabelste Betriebssystem der Welt 315
von Hubert Feyrer, Stefan Schumacher und Mark Weinem

Flüssiger Code – Open-Source-Software in der Medienkunst 327
von Martin Pichlmair

Teil III Open Source inspiriert

Kapitel 5 Open Source bildet

Open Source bildet 341
von Silvia Varadinova und Hans Henning

Einsatz von Open-Source-Software zur Vermittlung von IT-Schlüsselqualifikationen 343
von Christina Dörge

Pädagogische und gesellschaftliche Potenziale freier Software 351
von Herbert Reckmann

Zwei IT-Welten für alle SchülerInnen – Schule und freie Software. 365
von Christian Dorninger

Einsatz von Open-Source-Software in der Entwicklungszusammenarbeit – das Beispiel Afghanistan 377
von Nazir Peroz, Jan Suhr und Daniel Tippmann

Kapitel 6 Open Access

Kinderhände 393
von Gerhard Schöppe

Digitaler Open Access zu wissenschaftlichen Informationen 395
von Eberhard R. Hilf

Open Source, Open Content und Open Access – Gemeinsamkeiten und Unterschiede 413
von Reto Mantz

Open Access vs. DRM? 427
von Thomas Dreier und Kendra Stockmar

Ein Open-Content-Internetarchiv des Nachlasses eines zeitgenössischen Komponisten 439
von Kei Ishii

Open Fiction: Die neuen Ossis 453
von Joscha May

Kapitel 7 Im Dialog: Recht, Politik und Open Source

Die GPL bringt es auf den Punkt 459
von Martin Görnitz

Sharing – Ein Kampf ums Recht 461
von Bernd Lutterbeck

Die Zukunft elektronischer Demokratie 481
von Maika Büschenfeldt

Produkthaftung für Open-Source-Software? 495
von Christian Laux und Jan Widmer

Die GPL v3 – ein Schutzschild gegen das Damoklesschwert der Softwarepatente? 511
von Lina Böcker

Von der Kommune zur Community: Freie Netze und freies Wissen auf der lokalpolitischen Agenda 523
von Leonhard Dobusch

Glossar. 531

Stichwortverzeichnis 537

Mitwirkende 553

Sponsoreninformationen 561

Lizenzen 563

Vorwort der Herausgeber

Bernd Lutterbeck, Matthias Bärwolff
und Robert A. Gehring

(CC-Lizenz siehe Seite 563)

Vor Ihnen liegt die nunmehr vierte Ausgabe des Open Source Jahrbuchs. Wir hätten Grund genug, die Gelegenheit des Vorwortes zu nutzen, uns über alle Maße zu loben – unsere Arbeit und das Produkt, das Sie just in Ihren Händen halten. In Zeiten jedoch, da Leser sich mehr denn je durch die atemberaubende und von ihnen selbst geschaffene Transparenz ein eigenes Bild machen können, anstatt sich durch den Verkäufer ein Bild machen zu lassen, wirken solche Worte hohl und kalt. Sie erzählen keine Geschichte, sie wecken keine Emotionen, und nur selten sprechen sie überhaupt noch die Bedürfnisse ihrer Zielgruppen an. Im Zweifel erzeugen diese dann ihre Güter lieber selbst, und das in einem Miteinander, nicht Gegeneinander – ohne Marktgeschrei, ohne Gefeilsche und oft ohne Sachzwänge. Daher bleibt uns als Herausgebern nur, zuzugeben, dass wir in diesem Buch nur einen winzigen Ausschnitt der Realität abbilden können. Der Leser muss sich eben selbst ein Bild machen, und wenn wir mit diesem Buch einen winzigen Beitrag dazu leisten können, haben wir unser Ziel schon erreicht.

Die Vorstellung, dass ein Individuum überhaupt in der Lage sein kann, die Realität vollständig zu reflektieren und zu diskutieren, wurde spätestens in diesem Jahr von Norbert Elias' klassischem Postulat zerstört: Gerade weil ja eine Gesellschaft aus den sozialen Beziehungen ihrer Mitglieder entsteht und keines für sich existieren kann, hat der Katalysator Internet die sozialen Prozesse unserer Zeit so mächtig gemacht, selbst wenn sie in klassischen Marktprozessen aufgehen. Das Beispiel Open Source zeigt es in aller Deutlichkeit. Reizvoll sind sie zudem, wie *YouTube* und *MySpace* zeigen, und überdies vereinzelten am Markt gehandelten Prozessen kreativen Schöpfens zweifellos deutlich überlegen. Wahrheit entsteht eben in partizipativen kommunikativen Prozessen, nicht im Elfenbeinturm. Das einsame Genie der deutschen Romantik ist tot, es lebe das Miteinander – oder informatisch ausgedrückt: das „Sharing"!

Eben solches „Sharing" – der freie Austausch von Gedanken, Ideen und Texten – ist denn auch maßgeblicher Grund dafür, dass dieses Buch in der nunmehr vierten Ausgabe überhaupt einen Beitrag zur reichhaltigen und äußerst lebhaften Diskussi-

on um Open Source beitragen kann. In einer Redaktion, die sich überwiegend aus Studenten zusammensetzt, überrascht es kaum, dass es gelingen konnte, Autoren wie Richard Stallman oder Hal R. Varian für das Buch zu begeistern und ein Gesamtwerk zu komponieren, das die bisherigen Ausgaben deutlich in den Schatten stellt. Im Falle Stallmans hatten die Herausgeber schon vor drei Jahren das Handtuch geworfen, aber die Redaktion hat den Prozess der Kommunikation mit Stallman einfach erneut angestoßen, mit anderen Ideen und Vorstellungen, als wir sie vor drei Jahren hatten.

In der Tat sind die spannendsten Arbeiten und Errungenschaften des Projekts auf Initiative von Studenten hin entstanden: das Layout des Buches (Wolfram), die Gestaltung und technische Umsetzung der Webseite (Daniel), Strategien im Umgang mit Presse und Partnern (Bastian und Sebastian) und nicht zuletzt eben die inhaltliche Gestaltung des Buches (alle). Überall sind Dinge entstanden, die wir vor vier Jahren noch nicht einmal *gedacht* haben. Das Thema des Buches findet denn auch seine Entsprechung in der Form der Kommunikation im Projekt: Zur Arbeit der Redaktion gehören zuvorderst klassische Open-Source-Medien wie E-Mail-Liste und SVN-Repository. Der Diskurs, nicht nur innerhalb der Redaktion, sondern auch mit Autoren, ist lebendig und unvorhersehbar – Open Source im besten Sinne eben.

Zum Inhalt dieses Sammelbandes nur soviel: Trotz der thematischen Vielfältigkeit der Beiträge gliedert sich das Buch in diskrete semantische Fächer, die eine Orientierung im Buch hoffentlich erleichtern. Dennoch, jeder Artikel steht für sich, das Buch kann also von hinten nach vorne oder quer, ganz nach Belieben gelesen werden. Der Artikel von Richard Stallman eröffnet das Buch und steht außerhalb der Kapitelstruktur des Buches. Ebenso eine kurze Replik aus dem Kreis der Herausgeber. Noch etwas: Auch wenn sich dieses Buch „Jahrbuch" nennt, wir haben die Artikel derart gewählt, dass sie über den zeitlichen Rahmen dieses Jahres hinweg Relevanz und Gewicht haben werden. Die Artikel in den vorangegangenen Jahrbüchern sind ebenso wenig hinfällig, ein Blick zurück lohnt sich also.

Bevor wir schließen, möchten wir uns bedanken bei unserem Verlag Lehmanns Media, der Presseabteilung der TU Berlin, der TU Servicegesellschaft sowie Eva Affandi und ihren Studenten der HU Berlin, die uns beim Übersetzen einiger englischer Artikel ins Deutsche geholfen haben. In der studentischen Redaktion waren es insbesondere Matthias Liebig, Daniel Auener und Sebastian Stein, die dem Projekt gerade in der heißen Schlussphase enormen Halt gegeben haben. Zu guter Letzt dürfen wir Sie auf unsere im Laufe des Jahres völlig überarbeitete Webseite opensourcejahrbuch.de hinweisen, auf der Sie die Artikel aller Bücher, aber auch weiterführende Medien zum Thema Open Source finden. Hier können Sie die Artikel zudem kommentieren und bewerten. Auch für Ihre Meinung zum Buch im Allgemeinen, Anregungen und Wünsche sind wir Ihnen jederzeit dankbar.

Bernd Lutterbeck
Matthias Bärwolff
Robert A. Gehring

Warum „Open Source“ das Wesentliche von „Freier Software“ verdeckt*

VON DER FREIE-SOFTWARE-BEWEGUNG: RICHARD STALLMAN

*(Copyright R. Stallman**)*

1 Einleitung

Wenn wir von Freier Software sprechen, drücken wir damit aus, dass sie die unverzichtbaren Freiheiten der Anwender respektiert: die Freiheit, die Software laufen zu lassen; die Freiheit, sie zu untersuchen und zu ändern; die Freiheit, bearbeitete oder unbearbeitete Kopien davon weiterzugeben. Es geht uns um die Freiheit, nicht ums Geld! Denken Sie also an „Redefreiheit“ und nicht an „Freibier“.

Die genannten Freiheiten sind äußerst wichtig. Sie sind unverzichtbar – nicht nur im Interesse der einzelnen Anwender, sondern weil sie die Solidarität in der Gesellschaft fördern, das Miteinanderteilen und das gemeinschaftliche Handeln. Je mehr unsere Kultur und unsere Alltagsgeschäfte digitalisiert werden, desto größere Bedeutung gewinnen diese Freiheiten. In einer Welt digitaler Klänge, Bilder und Worte wird Freie Software immer mehr zum Synonym für Freiheit im Allgemeinen.

Viele Millionen Menschen setzen heute weltweit Freie Software ein. In einigen Regionen in Indien und Spanien lernen bereits alle Schüler den Umgang mit dem freien GNU/Linux-Betriebssystem. Doch die meisten dieser Anwender wissen nichts von den ethischen Gründen, aus denen wir dieses System entwickelt haben; sie haben nie etwas über die ethischen Motive unserer Gemeinschaft von Befürwortern der Freien Software gehört. Das liegt daran, dass das System und die Community immer häufiger mit dem Attribut „Open Source“ belegt und so mit einer Philosophie in Verbindung gebracht werden, in der diese Freiheiten kaum eine Rolle spielen.

---

* Aus dem Englischen übersetzt von Robert A. Gehring.

Anmerkung des Übersetzers: „Freie Software“ ist die wortwörtliche Übersetzung des englischen Begriffs *free software*. Der übersetzte Begriff wird im Artikel als Eigenname behandelt und dementsprechend groß geschrieben, ebenso bei „Open Source“. Anführungszeichen zur Hervorhebung der jeweiligen Begriffe wurden aus dem Originaltext übernommen.

** Copyright 2007 Richard Stallman. Die unveränderte Wiedergabe und Verbreitung dieses gesamten Artikels in beliebiger Form ist gestattet, sofern dieser Hinweis beibehalten wird.

Die Freie-Software-Bewegung setzt sich seit 1983 für die Freiheiten der Computeranwender ein. Im Jahr 1984 haben wir damit begonnen, ein freies Betriebssystem namens *GNU* zu entwickeln. *GNU* sollte die unfreien Betriebssysteme ersetzen, die den Anwendern ihre Freiheiten verwehrten. In den achtziger Jahren haben wir auf der einen Seite wichtige Systemkomponenten entwickelt und auf der anderen Seite die *GNU General Public License*. Die *GNU General Public License* wurde ausdrücklich dafür geschrieben, die Freiheiten aller Anwender eines Programms zu schützen.

Nicht alle Entwickler und Anwender Freier Software waren mit den Zielen der Freie-Software-Bewegung einverstanden. Einige von ihnen bildeten 1998 eine Splittergruppe und fingen an, für „Open Source" die Werbetrommel zu rühren. Am Anfang wurde der Ausdruck „Open Source" mit dem Argument vorgeschlagen, dass sich damit einige Missverständnisse in der Interpretation des Begriffs „Freie Software" vermeiden ließen. Aber schon recht bald verbanden sich damit philosophische Ansichten, die sich stark von denen der Freie-Software-Bewegung unterschieden.

Einige der Befürworter von „Open Source" sahen es als eine „Marketing-Kampagne für Freie Software". Durch die Betonung praktischer Vorteile und den Verzicht auf möglicherweise unbequeme Diskussionen darüber, was richtig oder falsch sei, sollte die Attraktivität für die Geschäftswelt erhöht werden. Es gab aber auch Open-Source-Verfechter, die die ethischen und sozialen Wertvorstellungen der Freie–Software-Bewegung schlicht ablehnten. Ganz egal, welche Position sie einnahmen, in ihrer Werbung für „Open Source" wurden diese Werte nicht erwähnt. Und so kam es, dass der Ausdruck „Open Source" nach kurzer Zeit in der Praxis nur noch für praktische Werte wie leistungsfähige, zuverlässige Software stand. Die meisten Open–Source-Unterstützer sind erst später dazugekommen und so ist es für sie ganz normal, den Begriff „Open Source" nur noch in diesem praktischen Sinn zu verstehen.

Fast jede Open-Source-Software ist Freie Software, die beiden Begriffe beschreiben so ziemlich dieselbe Kategorie von Software. Aber sie stehen für grundverschiedene Wertvorstellungen. Open Source ist eine Entwicklungsmethode, bei Freier Software geht es um eine soziale Bewegung. Für die Freie-Software-Bewegung ist Freie Software ein ethischer Imperativ, denn nur Freie Software respektiert die Freiheit des Nutzers. Im Gegensatz dazu fragt die Open-Source-Philosophie immer nur danach, wie man Software „besser" machen kann, also nach praktischen Aspekten. Die Kernaussage ist, dass unfreie Software eine suboptimale Lösung darstellt. Die Freie-Software–Bewegung sieht unfreie Software hingegen als ein soziales Problem an, für das Freie Software die Lösung darstellt.

Freie Software, Open Source; wenn es im Grunde um dieselbe Software geht, spielt es da überhaupt eine Rolle, welchen Namen man dem Kind gibt? Ja, denn unterschiedliche Wörter vermitteln unterschiedliche Ideen. Ein freies Programm mag Ihnen hier und heute auch unter anderem Namen dieselben Freiheiten einräumen. Um aber Freiheit zu schaffen, die von Dauer ist, muss man den Menschen den Wert

von Freiheit vermitteln. Wenn Sie dabei helfen wollen, kommen Sie nicht umhin, über „Freie Software" zu sprechen.

Wir, die Freie-Software-Bewegung, sehen das Open-Source-Lager nicht als Gegner an. Proprietäre Software ist der Gegner. Doch wir wollen, dass die Menschen verstehen, dass wir für Freiheit eintreten. Deshalb lassen wir uns nicht als Open–Source-Unterstützer bezeichnen.

2 Missverständnisse von „Freier Software" und „Open Source"

Der Ausdruck „Freie Software" kämpft mit einem Interpretationsproblem. In unserem Sinne bezieht sich das „frei" auf Freiheit, wie beispielsweise in der Phrase „Software, die dem Anwender gewisse Freiheiten gibt". Sprachlich korrekt, aber nicht in unserem Sinne lässt sich das „frei" auch als kostenlos interpretieren: „Software, die es kostenlos gibt". Wir lösen dieses Problem, indem wir eine Definition für „Freie Software" veröffentlichen und den Leuten sagen: „Denkt an Redefreiheit, nicht an Freibier." Das ist sicher keine perfekte Lösung; das Problem lässt sich damit nicht ganz aus der Welt schaffen. Ein unzweideutiger und korrekter Ausdruck wäre sicher besser, wenn es damit keine anderen Probleme geben würde.

Unglücklicherweise sind alle anderen, in der englischen Sprache zur Verfügung stehenden Alternativen auch nicht unproblematisch. Viele Leute haben uns Alternativen vorgeschlagen, und wir haben sie uns alle angesehen, aber keine von denen ist so offensichtlich „richtig", dass ein Wechsel eine gute Idee wäre. Jeder vorgeschlagene Ersatz für „Freie Software" hat seine eigenen semantischen Probleme – und das gilt auch für „Open-Source-Software".

Die offizielle Definition von „Open-Source-Software" (die von der *Open Source Initiative* veröffentlicht wird und zu lang ist, um sie hier zu zitieren) wurde indirekt von unseren Kriterien für Freie Software abgeleitet. Es gibt allerdings einige Unterschiede: In mancher Hinsicht ist sie liberaler, und das hat dazu geführt, dass die Open-Source-Unterstützer einige Lizenzen akzeptiert haben, die unserer Meinung nach die Anwender zu sehr einschränken. Davon abgesehen, kommen sich die beiden Definitionen in der Praxis ziemlich nahe.

Nichtsdestotrotz scheint der Ausdruck „Open-Source-Software" zu bedeuten: „Du kannst dir den Quellcode ansehen" – und die meisten Menschen scheinen ihn genau so zu verstehen. Damit kommt aber ein viel schwächeres Kriterium zum Einsatz als bei Freier Software, viel schwächer auch als in der offiziellen Definition von Open Source beschrieben. Damit werden auch viele Programme erfasst, die weder Freie Software noch Open Source sind.

Da nun die scheinbare offensichtliche Bedeutung von „Open Source" nicht die ist, die von ihren Befürwortern beabsichtigt worden ist, verstehen die meisten Menschen den Begriff am Ende falsch. Der Schriftsteller Neal Stephenson definiert „Open Source" beispielsweise so:

> „Linux ist 'Open-Source-Software', was bedeutet, dass jedermann Kopien des Quellcodes bekommen kann."

Ich denke nicht, dass er die „offizielle" Definition ablehnt oder bestreitet. Ich glaube, er hat einfach nur die Konvention der englischen Sprache benutzt, um eine Bedeutung für den Begriff zu finden. Der US-Bundesstaat Kansas hat eine ähnliche Definition veröffentlicht:

> „Setzen Sie Open-Source-Software (OSS) ein. OSS ist Software, deren Quellcode kostenlos und öffentlich verfügbar ist. Allerdings variieren die spezifischen Lizenzbedingungen im Hinblick darauf, was man mit diesem Code machen darf."

Die Open-Source-Leute reagieren auf so etwas mit einem Verweis auf ihre offizielle Definition, was aber bei ihnen nicht so effektiv ist wie bei uns. Der Ausdruck „Freie Software" hat zwei natürliche Bedeutungen, von denen eine die von uns verwendete ist. Wer einmal die Idee von „Redefreiheit, nicht Freibier" verstanden hat, weiß in Zukunft, was gemeint ist. Dagegen hat „Open Source" nur eine natürliche Bedeutung und diese unterscheidet sich von derjenigen, die ihre Befürworter ihr eigentlich zumessen. Und so kommt es, dass sich die offizielle Definition von „Open Source" nicht kurz und knapp erklären und begründen lässt. Die Verwirrung wird noch größer.

3 Unterschiedliche Wertvorstellungen können zu ähnlichen Schlussfolgerungen führen – aber das passiert nicht zwangsläufig

In den sechziger Jahren waren die radikalen Gruppen berühmt für ihren Hang zur Fraktionsbildung. Organisationen fielen im Streit um Strategiedetails auseinander und die daraus entstehenden Gruppen behandelten sich gegenseitig als Feinde, obwohl sie doch im Kern ähnliche Ziele und Wertvorstellungen hatten. Die Rechte hat das zu ihrem Vorteil benutzt und die ganze Linke heftig angegriffen.

Es gibt Leute, die versuchen, die Freie-Software-Bewegung schlecht zu machen, indem sie die Meinungsunterschiede zwischen uns und den Open-Source-Anhängern mit den Auseinandersetzungen innerhalb der radikalen Gruppen damals vergleichen. Das verdreht aber die Tatsachen. Die Unterschiede zwischen uns und dem Open–Source-Lager liegen bei den Zielen und Wertvorstellungen, aber unsere unterschiedlichen Ansichten führen in der Praxis oft zu demselben Verhalten – zum Beispiel zur Entwicklung Freier Software.

Das Ergebnis ist, dass Mitglieder der Freie-Software-Bewegung gemeinsam mit Vertretern aus dem Open-Source-Lager an Softwareprojekten arbeiten. Es ist schon bemerkenswert, wie derart unterschiedliche philosophische Standpunkte doch so oft

Menschen dazu motivieren können, zusammen an Projekten zu arbeiten. Und trotzdem sind die Auffassungen unterschiedlich und es gibt Situationen, in denen das zu unterschiedlichem Handeln führt.

Die Idee bei Open Source ist, dass Software leistungsfähiger und zuverlässiger wird, wenn man es den Anwendern erlaubt, die Software zu bearbeiten und weiterzuverbreiten. Dafür gibt es aber keine Garantie. Die Entwickler von proprietärer Software sind nicht notwendigerweise inkompetent. Manchmal schreiben Sie ein Programm, das leistungsfähig und zuverlässig ist, obwohl es die Freiheiten der Anwender nicht respektiert. Wie werden Aktivisten für Freie Software und Open-Source-Enthusiasten darauf reagieren?

Ein reiner Open-Source-Enthusiast, der überhaupt nicht von den Idealen der Freien Software beeinflusst ist, wird sagen: „Ich bin zwar überrascht, dass du so ein anständiges Programm produziert hast, ohne unser Entwicklungsmodell zu benutzen, aber du hast es geschafft. Wo bekomme ich eine Kopie her?" Durch solches Verhalten werden Handlungsweisen belohnt, die unsere Freiheiten beschneiden und schließlich zu deren Verlust führen.

Der Freie-Software-Anhänger wird sagen: „Dein Programm ist zwar attraktiv, aber nicht für den Preis meiner Freiheit. Ich werde daher ohne das Programm auskommen müssen. Stattdessen werde ich ein Projekt unterstützen, das eine freie Alternative entwickelt." Wenn uns unsere Freiheit etwas Wert ist, haben wir die Möglichkeit, so zu handeln, dass wir sie bewahren und verteidigen.

4 Leistungsfähige, zuverlässige Software kann schlecht sein

Unser Wunsch nach leistungsfähiger und zuverlässiger Software basiert auf der Grundannahme, dass Software dazu da ist, dem Anwender zu nützen. Je leistungsfähiger und zuverlässiger sie ist, desto besser kann sie seinen Zwecken dienen. Aber Software dient nur dann den Zwecken ihrer Anwender, wenn sie ihre Freiheit respektiert. Was aber, wenn die Software nur dazu dienen soll, den Anwendern Fesseln anzulegen? Dann bedeutet mehr Zuverlässigkeit nichts anderes, als das es schwerer wird, die Fesseln zu sprengen.

Unter dem Druck der Film- und Musikunternehmen wird Anwendersoftware zunehmend so gestaltet, dass sie die Handlungsspielräume der Anwender einschränkt. Dieses bösartige Verhalten ist bekannt unter dem Namen *DRM*, oder auch *Digital Restrictions Management*[1], und bildet die Antithese zu den Freiheitsidealen der Freie–Software-Bewegung. Und es geht nicht nur um Ideale: Da es das Ziel von *DRM* ist, Ihre Freiheiten zu beschneiden, bemühen sich die DRM-Entwickler darum, es Ihnen so schwer wie möglich zu machen, unmöglich oder sogar illegal, DRM-Software zu verändern.

1 Siehe http://www.DefectiveByDesign.org.

Trotzdem haben einige Open-Source-Befürworter den Vorschlag gemacht, „Open–Source-DRM-Software“ zu entwickeln. Ihr Vorschlag läuft darauf hinaus, dass durch die Veröffentlichung der Quellcodes von Programmen, die dazu gedacht sind, den Zugang zu verschlüsselten Medien zu beschränken, eben diese Programme leistungsfähiger und zuverlässiger werden – Programme, die Anwender wie Sie in ihrem Handeln einschränken. Anschließend werden diese Programme zusammen mit Geräten ausgeliefert, die es Ihnen nicht mehr erlauben, die Software zu ändern.

Solche Software mag „Open Source“ sein, und sie mag das Open-Source-Entwicklungsmodell benutzen, aber sie wird keine Freie Software mehr sein, denn sie respektiert die Freiheiten der Anwender nicht mehr. Sollte das Open-Source-Entwicklungsmodell tatsächlich dazu führen, dass solche Software leistungsfähiger und zuverlässiger darin wird, Ihre Handlungsfreiheiten zu beschränken, macht das die ganze Sache nur schlimmer.

5 Die Angst vor der Freiheit

Der wichtigste Grund für die Verwendung des Begriffs „Open Source“ war, dass die ethischen Ideen von „Freier Software“ bei manchen Leuten ein Unwohlsein auslösen würden. Das stimmt: Über Freiheit, Ethik, Verantwortlichkeit und Bequemlichkeit zu reden, heißt, von den Leuten zu verlangen, dass sie über Dinge nachdenken, die sie lieber ignorieren würden, zum Beispiel, ob ihr Verhalten ethisch ist. Das kann Unbehagen auslösen und manche Leute werden sich dagegen sperren. Daraus folgt aber nicht, dass wir aufhören sollten, über diese Dinge zu sprechen.

Die Anführer der Open-Source-Bewegung haben sich aber entschieden, genau das zu tun. Sie haben gemerkt, dass sich ihre Software besser „verkaufen“ lässt, besonders an Unternehmen, wenn sie aufhören, über Ethik und Freiheit zu reden und stattdessen nur noch über den praktischen Nutzen der Software.

An seinem Ziel gemessen, hat sich dieser Ansatz als wirkungsvoll erwiesen. Die Rhetorik von Open Source hat viele Unternehmen und Individuen dazu gebracht, Freie Software zu nutzen oder sogar zu entwickeln, wodurch unsere Gemeinschaft gewachsen ist – oberflächlich gesehen, in rein praktischer Hinsicht. Die Philosophie von Open Source, mit ihrer Ausrichtung auf rein praktische Werte, steht einem tieferen Verständnis der Werte von Freier Software im Wege. Sie bringt zwar viele neue Mitglieder in unsere Gemeinschaft, lehrt sie aber nicht, diese auch zu verteidigen. Auch wenn neue Mitglieder keine schlechte Sache sind, so reicht das nicht aus, um die Freiheit auf ein sicheres Fundament zu stellen. Anwender dazu zu bringen, Freie Software zu verwenden, ist nur ein Schritt auf dem Weg, der sie schließlich dazu bringen soll, ihre eigene Freiheit auch zu verteidigen.

Früher oder später werden diese Anwender mit Angeboten an proprietärer Software konfrontiert, die praktische Vorteile bietet und sie wieder zum Wechseln veranlassen soll. Zahllose Softwarehersteller locken mit solchen Angeboten, zum Teil sogar kos-

tenlos. Warum sollten die Anwender das zurückweisen? Dazu werden sie nur bereit sein, wenn sie den Wert der Freiheit, den Freie Software ihnen bietet, als solchen schätzen gelernt haben und ihm größere Bedeutung zumessen als den technischen und praktischen Bequemlichkeiten, die ihnen eine spezifische Freie Software bietet. Um das zu erreichen, müssen wir über Freiheit sprechen. In gewissem Umfang kann der „Stillschweigen-Ansatz" der Gemeinschaft nützlich sein, aber er wird ihr gefährlich, sobald er allgegenwärtig wird und die Liebe zur Freiheit nur noch als exzentrisch wahrgenommen wird.

Genau in dieser gefährlichen Situation stecken wir gegenwärtig. Die meisten Menschen, die etwas mit Freier Software zu tun haben, reden nicht viel über Freiheit, weil sie so in der Geschäftswelt leichter Akzeptanz finden. Dieses Verhaltensmuster kann man besonders bei den Softwaredistributoren beobachten. Praktisch alle GNU/Linux-Distributoren kombinieren das freie Betriebssystem mit proprietären Softwarepaketen. Gegenüber den Anwendern stellen sie als Vorteil dar, was eigentlich einen Verlust an Freiheit bedeutet.

Proprietäre Softwarepakete und zum Teil unfreie GNU/Linux-Distributionen fallen auf fruchtbaren Boden, da die meisten Mitglieder unserer Gemeinschaft nicht mehr auf den Softwarefreiheiten bestehen. Das ist kein Zufall. Die meisten GNU/Linux–Anwender sind zu dem System über eine „Open-Source-Diskussion" gekommen, die Freiheit nicht als Ziel propagiert. Handlungsweisen, die die Freiheit nicht verteidigen, und Reden, in denen Freiheit nicht vorkommt, gehen Hand in Hand, bestätigen sich wechselseitig. Um diese Entwicklungstendenz zu überwinden, müssen wir häufiger, nicht seltener, von Freiheit reden.

6 Zusammenfassung

Zu einer Zeit, in der die Befürworter von Open Source unserer Gemeinschaft immer neue Mitglieder zuführen, müssen wir als Freie-Software-Anhänger uns noch stärker dafür einsetzen, bei diesen neuen Mitgliedern Bewusstsein für die Sache der Freiheit zu schaffen. „Es ist Freie Software und sie gibt dir Freiheit!" – Das müssen wir lauter und öfter sagen als zuvor. Und jedes Mal, wenn Sie „Freie Software" an Stelle von „Open Source" sagen, unterstützen Sie unsere Sache.

Die ökonomischen Grenzen freier Software

MATTHIAS BÄRWOLFF

(CC-Lizenz siehe Seite 563)

1 Die Freiheit und ihre Grenzen

Wenn die Freiheit des Einzelnen und die prinzipielle Unverletzlichkeit des Eigentums das Fundament unserer Gesellschaft bilden sollen, so gehört dazu zweifellos auch die Freiheit, anderen sein Eigentum oder Rechte daran in freundlicher Absicht weiterzugeben. Die Freiheit, von der Richard Stallman in seinem Artikel „Warum 'Open Source' das Wesentliche von 'Freier Software' verdeckt" spricht, hat also wahrlich nichts mit Kommunismus zu tun, sondern mit genau den bürgerlichen Freiheiten, die wir auch den Ackermanns dieser Welt zubilligen.

Eben dieser Freiheiten bemächtigte sich Stallman in seiner berühmten *GNU General Public License (GPL)*, dem institutionellen Fundament freier Software, mit dem er eine der wohl weitreichendsten Innovationen im Bereich der Softwareproduktion, -Verbreitung und -Nutzung schuf. Heute bezweifelt niemand ernsthaft, dass freie Software und entsprechende Lizenzmodelle als soziale Praktiken dem gesellschaftlichen Normen- und Rechtssystem unserer Gesellschaft entsprechen. Alle Versuche, freier Software ihre Existenzberechtigung abzusprechen, sei es mit ökonomischen, rechtlichen oder gar moralischen Argumenten, sind bislang noch gescheitert. Und es bleibt zu hoffen, dass sich Politik und Gerichte weiterhin der Argumente derer erwehren können, denen aus freier Software eine kommerzielle Bedrohung erwächst. Die Macht und das Versprechen von freier Software liegen nämlich einerseits in ihrer Freiheit, andererseits aber auch in ihrer unübersehbaren kommerziellen Bedeutung mithin volkswirtschaftlichen Wertschöpfung.

Und hier beginnt denn auch der Konflikt, den Stallman beschreibt – zwischen Freiheit auf der einen Seite und „technischen und praktischen Bequemlichkeiten" auf der anderen Seite. Für ihn ist die Sache eindeutig: Die Freiheit von Software ist ein Wert, der sich dem ökonomischen Kalkül entzieht und deren Verlust – auch teilweise – so schwer wiegt, dass keine Abstriche in Kauf zu nehmen sind. So postuliert er: „Software dient nur dann den Zwecken ihrer Anwender, wenn sie ihre Freiheit

respektiert.“ Paradoxerweise sind es nun die Anwender selbst, die noch von diesem Argument überzeugt werden müssen. Stallman bemerkt treffend:

> „Warum sollten die Anwender [günstige proprietäre Software mit praktischen Vorteilen] zurückweisen? Dazu werden sie nur bereit sein, wenn sie den Wert der Freiheit, den Freie Software ihnen bietet, als solchen schätzen gelernt haben und ihm größere Bedeutung zumessen als den technischen und praktischen Bequemlichkeiten, die ihnen eine spezifische Freie Software bietet.“

Der normative Gestus und damit die Anfälligkeit einer solchen Position sind unübersehbar. Der Wert der Freiheit als solche ist für Stallman so fundamental, dass er sämtliche praktische Erwägungen seitens des Anwenders verdrängt, ja ob seiner Absolutheit verdrängen muss. Für die meisten Computer-Anwender hingegen hat Software letztlich nur wenig mit Freiheit zu tun, sondern vielmehr *nur* mit praktischen Erwägungen, die noch dazu in Konkurrenz mit anderen Erwägungen stehen. Die Freiheit einer Software ist für die meisten ein untergeordnetes Argument in ihrem persönlichen Nutzenkalkül. Mithin folgt die Freiheit des Einzelnen nur mittelbar, wenn überhaupt, aus der Freiheit einer Software. Es ist ja sogar so, dass die Freiheit der Software nur um den Preis der Beschneidung individueller Freiheiten gewonnen werden kann. Der Anwender darf mit einer freien Software nämlich genau die Dinge nicht tun, die die Freiheit der Software einschränken würden.

Kurzum, mit der *GPL* wird zwar die unbedingte Freiheit der Software durchgesetzt, nicht aber die der Anwender. Die *GPL* beschneidet die Freiheiten des Einzelnen zugunsten einer abstrakteren Freiheit, die ihre Bedeutung erst im Miteinander der Individuen findet. Den Wert dieser Freiheit aufzuzeigen, diese Freiheit und ihren Nutzen spürbar gemacht zu haben durch seine Taten – darin liegt das große Verdienst Stallmans. Ob sich diese Freiheit jedoch gegenüber anderen Erwägungen behaupten kann, entscheidet sich letztlich bei den Anwendern und deren Nutzenkalkülen.

2 Freiheit versus Bequemlichkeit

Open Source gewichtet Freiheit pragmatischer als freie Software. Die Kriterien freier Software, die die Freiheit der Software sicherstellen sollen, sind bei Open Source deutlich schwächer ausgeprägt. Wenn wir die Freiheit der Software nicht zum Selbstzweck erheben, wundert es wenig, dass die Abwägung zwischen Freiheit und „technischen und praktischen Bequemlichkeiten“ eher zugunsten Letzterer ausfällt. Für Open Source ist die Freiheit der Software nur Mittel zum Zweck. Die Freiheit des Einzelnen im Umgang mit der Software steht im Vordergrund, auch wenn dadurch die Freiheit der Software selbst eingeschränkt wird. Man denke an die *Berkeley Software Distribution License (BSD)*, die es dem Benutzer erlaubt, die Software in praktisch beliebiger, auch proprietärer Weise zu nutzen. Was wir dadurch verlieren und gewinnen sollte eine empirische Frage sein, keine kategorische.

Fraglos lassen Open Source und die pragmatische Einstellung gegenüber der Freiheit Systeme zu, die dem Gedanken der Handlungsfreiheit der Nutzer diametral widersprechen. In diesem Punkt ist Stallmans Kritik an Digital-Rights-Management–Systemen (DRM) zuzustimmen. Tatsächlich gibt es auch im „Open-Source-Lager" genügend Befürworter der *GPL v3*, die DRM-Systemen deutlich ablehnend gegenüberstehen. Auch lässt Open Source es zu, dass Verbesserungen an einer Software nicht wieder, wie unter der *GPL*, der Öffentlichkeit zur Verfügung gestellt werden müssen. Andererseits sind die Errungenschaften und die Wertschöpfung der Open–Source-Community inklusive all der Firmen, die darin tätig sind, auch derer, die Open-Source-Lösungen für ihre proprietären Produkte verwenden, unübersehbar.

Der entscheidende Punkt ist letztlich dieser: Eine ökonomisch motivierte Abwägung zwischen Freiheit und „technischen und praktischen Bequemlichkeiten" kann sowohl für den Einzelnen als auch die Allgemeinheit bessere Resultate erzielen als der kategorische Verzicht darauf. Hierfür ist es aber unabdingbar, den Absolutheitsanspruch der Freiheit zu lockern.

3 Kommerz und Wertschöpfung

Das klassische Problem der Ökonomie ist es, existierende Ressourcen derart zu verteilen, dass sie ihren höchsten Nutzen entfalten. Hierfür gibt es prinzipiell drei Möglichkeiten: Markt, Firma und Allmende (Coase 1960). Selbiges gilt für die Schöpfung neuer Ressourcen und der wertschöpfenden Rekombinierung existierender Ressourcen. Auch hier hängt es von den jeweiligen Transaktionskosten ab, welche der drei Governance-Strukturen die effizientere ist.

Es lässt sich nun im Bereich Software, wie auch in vielen anderen Bereichen, unterscheiden zwischen der Schöpfung auf der einen Seite und der Anwendung auf der anderen Seite.[1] Der entscheidende Punkt bezüglich der Governance-Strukturen ist, dass sich die Transaktionskostenstrukturen in beiden Bereichen unterscheiden. Im Bereich der Entwicklung von komplexer Software kann es tatsächlich ökonomisch sinnvoll sein, diese als ein Allmendegut zu behandeln (Bessen 2001). Wo es aber um die Befriedigung spezifischer Endanwenderbedürfnisse geht, sinken die relativen Vorteile eines Allmende-Regimes. Hier ist es insbesondere nicht mehr möglich, sämtliche an dieser Schnittstelle stattfindenden Transaktionen ohne formale Verträge und Geldzahlungen abzuwickeln. Dies ist effizient dort, wo informelle Communitys mit effektiven Mechanismen zur Durchsetzung sozialer Normen existieren (Bowles 2004, S. 258), nicht aber an Schnittstellen zwischen Produzenten und Konsumenten,

1 Auch wenn es für Anwender prinzipiell möglich ist, Software selbst weiterzuentwickeln und ihren Bedürfnissen anzupassen, überwiegen die Kosten hierfür im Allgemeinen den Nutzen, so dass diese Möglichkeit für viele Endanwender auch bei Freier Software oder Open Source eine rein theoretische bleibt. Die Trennung zwischen Produzenten und Konsumenten, Arbeitsteilung und Spezialisierung also, bleibt trotz der „Demokratisierung von Innovation" (von Hippel 2005) effizient, schon allein wegen der Komplexität von Software.

die normalerweise von konventionellen geldvermittelten Markttransaktionen dominiert werden. Wenn hier also Software weitergegeben wird, die nicht vollständig frei ist, sondern einfach nur die Bedürfnisse des Anwenders befriedigt und sich dadurch effizientere Anreizstrukturen für die Produktion schaffen lassen, mag dies dem Ideal der unbedingten Freiheit von Software zuwiderlaufen, ist jedoch die ökonomisch sinnvollere Alternative. Mithin ist die Bedeutung von strikt kommerziellen Anreizen in der Produktgestaltung nicht zu unterschätzen. Auch wenn Bruce Perens in seinem Artikel in diesem Buch (S. 131) richtig bemerkt, dass auch Marketingabteilungen „keine Kristallkugel" haben, so übersieht er jedoch den entscheidenden Transaktionskostenaspekt bei der Wahl der Institutionen, wenn er postuliert, dass „das auf dem offenen Markt basierte Open-Source-Prinzip in der Schaffung wünschenswerter Produkte bessere Erfolge erzielen kann als zentral geleitete Marketing-Abteilungen und ihr Management". Denn Marketingabteilungen können sehr wohl Effizienzen in der Kommunikation zwischen Produzenten und Konsumenten erzielen, insbesondere dort, wo Trennung zwischen beiden eben scharf ist. Vor allem haben sie einen konkreten kommerziellen Anreiz, die Bedürfnisse von Anwendern, nicht Programmierern, tatsächlich zu bedienen – auch wenn, ja gerade weil dies mit Risiken und Kosten einhergeht. Ein solcher Anreiz mag zwar auch im Allmende-Modell vorhanden sein, jedoch ist er ob der informellen Vertragsstrukturen und der weitgehenden Abwesenheit von direkten Geldanreizen nicht so stark ausgeprägt. Die Behauptung, die potenziell unendlichen Permutationen in der Allmende würden nach „darwinistischer Selektion" zwangsläufig zu besserer Software führen, ist empirisch dünn.

Kurzum, Stallmans Inversion der konventionellen Eigentumslogik mit dem Zweck, eine Allmende zu schaffen, die den Schutz der Gerichte genießt, ist die wohl bedeutendste institutionelle Innovation der letzten Jahrzehnte. In einer breiteren Auslegung ist sie als Meta-Idee im Sinne von Romer (1990) zu verstehen. Dennoch, die Allmende kann und will nicht allen Gegebenheiten der Realität gerecht werden. Die Balance zwischen den Polen zu finden, ist wie so häufig die Herausforderung, vor der wir beständig stehen. Es wundert aber kaum, dass die erfolgreichsten Modelle in der Softwareindustrie Hybriden aus Freiheit und Nicht-Freiheit sind.

Literatur

Bessen, J. E. (2001), 'Open Source Software: Free Provision of Complex Public Goods', *Research on Innovation* **7**(2).

Bowles, S. (2004), *Microeconomics: Behavior, Institutions, and Evolution*, Princeton University Press.

Coase, R. H. (1960), 'The problem of social cost', *Journal of Law and Economics* **3**, S. 1–44.

Romer, P. M. (1990), 'Endogenous Technological Change', *Journal of Political Economy* **98**(5), S. 71–102.

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

Kapitel 1

Das Prinzip Open Source

„Ich habe eiserne Prinzipien. Wenn sie Ihnen nicht gefallen, habe ich auch noch andere."

– Groucho Marx

Das Prinzip Open Source: Einleitung

DANIEL AUENER

(CC-Lizenz siehe Seite 563)

„Es ist Freie Software und sie verschafft dir Freiheit!“, schreibt Richard Stallman in seinem Beitrag für dieses Jahrbuch. Er und die Freie-Software-Bewegung werben dafür, statt Open Source den Begriff freie Software zu verwenden. In Zeiten, in denen Open Source für Management, die großen IT-Abteilungen und Politiker kaum noch ein Fremdwort ist, scheint der Disput um die Begriffe *freie Software* und *Open Source* fast etwas müßig. Denn Open Source hat sich längst durchgesetzt. Immer mehr Firmen verfolgen profitable Geschäftsmodelle basierend auf Open Source. *Red Hat* oder *Novell* etwa erzielen rund um ihre Linux-Distributionen enorme Gewinne und sind im Nasdaq-Index vertreten. Darüber hinaus wird Open Source tatkräftig von der Politik unterstützt: von der brasilianischen Regierung, die sich eindeutig gegen proprietäre Software ausspricht, bis hin zur Münchener Stadtverwaltung, dem Deutschen Bundestag oder dem Auswärtigen Amt, die alle große Migrationsprojekte gestartet bzw. abgeschlossen haben. Auch die Tagespresse und eine relativ breite Öffentlichkeit kommen um das Thema lange nicht mehr herum.

Open Source, als Methode der Herstellung und des Vertriebs von Software, stellt zweifellos eine der wichtigsten institutionellen Innovationen in der IT-Branche dar. Trotzdem gibt es Grund, Stallmans Worte nicht ungehört verklingen zu lassen. Erfolg ist gut, aber er kann auch zu übereilter Legitimation führen. Gerade wenn man aus dem rasanten Aufstieg von Open Source lernen möchte oder sogar in anderen Bereichen als der Softwareentwicklung mit denselben oder ähnlichen Methoden Erfolg haben möchte, ist es wichtig, sich der Geschichte und der Wurzeln von Open Source bewusst zu sein. Selbst wenn man die kompromisslose Haltung Stallmans und der *Free Software Foundation* zur Bezeichnung „Open Source“ nicht teilt und sich nicht als Teil einer politischen Bewegung zur Befreiung des Quelltextes sieht, kann man ihnen doch zumindest in einem Punkt zustimmen: Open Source ist nicht nur ein cleveres Geschäftsmodell, nicht nur eine effiziente Entwicklungsmethode und nicht nur Triebfeder neuer Innovationen. Hinter Open Source steht auch eine Idee, es gibt ein *Prinzip Open Source*, das über die Probleme der Softwareentwicklung hinaus bedeu-

tend ist und in seiner Begrifflichkeit gerade erst in der wissenschaftlichen Diskussion erfasst wird.

Um dieses Prinzip geht es im folgenden Kapitel. Dabei fassen der erste und der letzte Artikel von Gundolf Freyermuth bzw. Franz Nahrada dieses Kapitel in einen chronologischen Rahmen. Freyermuth kommt nach seiner historischen Betrachtung der Open-Source-Praxis zu der Einsicht, dass nicht nur die Softwareentwicklung, sondern unsere ganze Gesellschaft momentan an einem Wendepunkt steht: am Anfang eines Zeitalters – ähnlich dem der industriellen Revolution –, das unser Denken, unsere Arbeits- und Lebensweise völlig ändern könnte. Franz Nahrada dagegen geht den großen Schritt, den wir nach Freyermuth gerade begonnen haben, in Gedanken zu Ende und schaut auf eine zukünftige Gesellschaft der „globalen Dörfer". Die Strukturen, die er beschreibt, könnten entstehen, wenn die Gesellschaft den Open–Source-Gedanken konsequent zu Ende denkt, verinnerlicht und umsetzt.

Die Artikel im Mittelteil des Kapitels untersuchen das Prinzip Open Source in der Gegenwart. Alexander Knorr zeigt mit dem Konzept der kulturellen Aneignung einen ethnologischen Weg des Verständnisses auf. Möglichkeiten zur Aneignung, Umdeutung und Umarbeitung sind für ihn wesentliche Voraussetzung für die kreative und innovative Entwicklung einer Kultur, wobei „der Grad der Deutungsoffenheit und der Umarbeitbarkeit" eine entscheidende Rolle spielt.

Im Anschluss daran sucht Janet Hope Wege für die tatsächliche Übertragung des Open-Source-Gedankens auf ein anderes Gebiet, und zwar die Erforschung und Entwicklung von Medikamenten. Hierfür erarbeitet sie die grundlegenden Eigenschaften von Open-Source-Lizenzen, die es gilt, auf ähnliche Lizenzen für den Arzneimittelmarkt zu übertragen, will man verhindern, dass der Open-Source-Gedanke im pharmazeutischen Umfeld nicht verfremdet wird.

In einer Reflexion und Diskussion erfolgreicher Anwendungen des Prinzips Open Source beschreiben Frank Piller, Ralf Reichwald und Christopher Ihl zukunftsweisende Geschäftsmodelle, wie die „Open-Source-Poduktion" von T-Shirts oder Sportartikeln. Sie widmen sich den Chancen und Grenzen des Konzeptes der „interaktiven Wertschöpfung", das allgemein eine Produktion mit Open-Source-Methoden meint.

Alle Artikel dieses Kapitels zeigen die Dynamik, die entstehen kann, wenn man sich auf den Open-Source-Gedanken einlässt. Sie zeigen neue Wege, spannende Ansätze und wecken Interesse an zukünftigen Entwicklungen. Und sie machen eins deutlich: Das Prinzip Open Source bedeutet tatsächlich mehr Freiheit, z. B. für alle an einem Produktionsprozess Beteiligten. Diese Freiheit ist es, die wildfremde Menschen konstruktiv zusammenarbeiten lässt, zur Kreativität anregt und Innovationen hervorbringt. Sie gilt es zu bewahren, wenn man das Prinzip Open Source auch in anderen Bereichen als der Softwareentwicklung anwenden will.

Offene Geheimnisse – Die Ausbildung der Open-Source-Praxis im 20. Jahrhundert*

GUNDOLF S. FREYERMUTH

(CC-Lizenz siehe Seite 563)

Die Open-Source-Praxis wirkt als treibende Kraft beim Übergang von der industriellen zur digitalen Zivilisation. Ihr Ursprung liegt im Bedürfnis nach Selbstbestimmung im Gebrauch digitaler Technologie, das um 1960 in den Kreisen avantgardistischer US-Wissenschaftler entstand. Während der siebziger und achtziger Jahre bildeten sich zentrale praktische Verfahrensweisen aus. Mit der Eröffnung des WWW kam es in den neunziger Jahren zur globalen Popularisierung und zur Prägung des Begriffs Open Source selbst. Kennzeichen der innovativen Praxis sind die Selbstermächtigung der Nutzer, die Evolution von Standards, die Entwicklung egalitärer Produktionsgemeinschaften, die Kodifizierung von geistigem Gemeineigentum, vernetztes Wissensarbeiten und konkurrierende Kollaboration. In ihrer aktuellen Gestalt bietet die Open-Source-Praxis nicht nur Produktivitäts- und Wettbewerbsvorteile. Sie räumt dem Einzelnen auch neue Freiheit ein, motiviert in höherem Maße und befähigt zu kreativer Partizipation.

Schlüsselwörter: geistiges Gemeineigentum · konkurrierende Kollaboration · digitale Ordnung · Globale Vernetzung

1 Einleitung

Mehrere Jahrzehnte nachdem sie begann, geriet die globale Vernetzung Mitte der neunziger Jahre plötzlich ins öffentliche Bewusstsein – als reine Geschäftsgelegenheit.

* Der Text basiert auf mehreren Vorträgen, die ich in den vergangenen Jahren zur Digitalisierung und Open-Source-Praxis hielt, unter anderem 2001 am Zürcher Gottlieb Duttweiler Institut und 2003 im Rahmen der Kritischen Universität Kaiserslautern, sowie auf folgenden Einzelpublikationen: Freyermuth (2001*a*,*c*,*d*, 2003). Diese Arbeiten wiederum stützten sich auf folgende Publikationen: DiBona et al. (1999); Wayner (2000); Himanen (2001); Lessig (2001); Moody (2001); Raymond (1999); Torvalds und Diamond (2001).

Das Schlüsselereignis, mit dem das Internet populär und vom Kommunikationsraum der wissenschaftlich-technischen Elite zu einem Medium der Massen wurde, datiert auf den August 1995. Paul Ceruzzi (1998, S. 304) schrieb: „Reading the newspaper accounts and watching the television news, one had the feeling that the day *Netscape* went public marked the beginning of the history of computing, and that everything else been a prologue." Der raketengleiche Aufstieg der *Dotcoms*, für den *Netscapes* Erfolg damals den Startschuss gab, und die nicht minder spektakulären Bruchlandungen fünf, sechs Jahre später verdecken seitdem im breiten Bewusstsein eine simple Wahrheit: dass es keineswegs mehr oder minder geschickte Geschäftsleute waren, die in der zweiten Hälfte des 20. Jahrhunderts das technologische Fundament digitaler Vernetzung legten. Treibende Kraft waren vielmehr Tausende technisch begeisterter Männer (und wenige Frauen): zivile und militärische Forscher, programmierende Professoren und Studenten, engagierte Techniker. Ähnlich den Hightech-Bastlern, die in den 1970er Jahren gegen das hinhaltende Desinteresse der Computerhersteller den PC konstruierten, agierten die Pioniere digitaler Vernetzung seit den 1960er Jahren ohne Unterstützung, ja gegen den Widerstand der monopolistischen Telekommunikationskonzerne.[1] Die Produkte ihrer Brillanz ließen sich zudem die Akademiker und Hobbyisten gerade nicht im Interesse profitorientierter Verwertung patentieren. Stattdessen stellten sie über Jahrzehnte hinweg das, was sie schufen gratis und samt Quellcode zur Verfügung. So sorgten sie dafür, dass ihre Programme für andere nicht nur nutzbar, sondern les- und veränderbar waren. Diese unzähligen und größtenteils namenlos gebliebenen digitalen Handwerker und Bastler, denen es primär nicht um Tausch-, sondern Gebrauchswert ging, schrieb Jon Katz, waren „die wahren Architekten der *new economy*".[2]

Im offenen, durch kein Copyright behinderten Austausch der Experten entstand so sukzessive die Softwarebasis digitaler Kultur. Auf ihr – kollektiven, dem Gemeinbesitz überantworteten Kreationen – beruhen bis in die Gegenwart die wichtigsten Programme, die das Internet am Laufen halten, unter anderem *Sendmail*, *BIND*, *Apache*, *Perl* und *Python*. Die Bedeutung von Offenheit demonstriert das Internet zudem über reine Software hinaus. *TCP/IP*, das offene Basisprotokoll des Netzes, konnte proprietäre Protokolle wie *IPX* oder *NetBEUI* fast völlig verdrängen. Die meisten Standards und Protokolle liegen als *RFCs* (*Requests for Comments*)[3] vor und verdanken ihre Existenz öffentlichen Diskussionen.[4]

Nicht mit allen dieser Namen und Kürzel wird jeder auf Anhieb deren exakte

1 Zum Widerstand der Computerhersteller gegen den PC vgl. z. B. Levy (1984, S. 184 ff.); zum Widerstand der Telekommunikationskonzerne vgl. z. B. Abbate (1999, S. 135) und Frank (1990, S. 21).

2 Zitat auf dem Schutzumschlag von Himanen (2001).

3 Zum Request-for-Comments-Verfahren (*RFC*) siehe Abschnitt 6. Vgl. auch das Stichwort *RFC* im *Free Online Dictionary of Computing* http://foldoc.org/index.cgi?query=rfc&action=Search oder auf der Living-Internet-Webseite http://www.livinginternet.com/i/ia_rfc_invent.htm.

4 Auch das heute zentrale *HTTP* entwickelte Tim Berners-Lee in einem offenen Prozess, der viele Beiträger hatte. Vgl. dazu Abschnitt 7.

Funktion und vor allem deren genaue Entstehungsgeschichte verbinden können. Die Details sind freilich in diesem Zusammenhang nicht entscheidend. Es reicht, sich ihre grundsätzliche Bedeutung zu vergegenwärtigen, etwa in einem bekannten Vorbild analoger Technik: Gliche der digitale Datenverkehr der Eisenbahn, so erfüllten Weichen, Stellwerke, Signale und Fahrpläne ähnliche Funktionen.

Gar nichts also ginge ohne nonkommerzielle Software. Das Programm *Sendmail* etwa läuft auf 80 Prozent aller Mailserver, kaum eine elektronische Post, die auf ihrem verschlungenen Weg vom Sender zum Empfänger nicht mit diesem Gratis-Code in Berührung käme. Ebenso abhängig sind Surfen im *WWW* und E-Commerce von freier Software: Zu Beginn des 21. Jahrhunderts waren 59,95 Prozent aller Webserver freie *Apaches*, während die gesammelten Closed-Source-Angebote von *Microsoft* mit 25,8 Prozent auf den zweiten Platz verwiesen wurden. An diesem Verhältnis hat sich, während das Netz und der Verkehr in ihm um ein Vielfaches wuchsen, bis heute nur wenig geändert.[5] Mehr noch bedroht Linux die Dominanz des Redmonder Softwarekonzerns: Binnen eines Jahrzehnts stieg, was in den frühen neunziger Jahren mit 10 000 Programmzeilen eines Studenten begonnen hatte, zu einem Kollektivschatz von mehreren 100 Millionen Codezeilen auf – allein der Betriebssystemkern von Linux 2.6.0 umfasst rund sechs Millionen Zeilen. Nach den üblichen Tarifen der Branche stellen sie einen Investitionswert von vielen Milliarden Dollar dar.[6]

Microsoft erkannte die Open-Source-Praxis denn auch früh als grundsätzliche Gefahr. Die zügig einsetzenden Abwehrversuche zeugen davon: die berüchtigten *Halloween Papers*, die 1998 an die Öffentlichkeit gerieten, ebenso wie Microsoft-CEO Steve Ballmers Reden, in denen er Linux im Jahr 2001 als „Bedrohung Nr. 1" und als „Krebsgeschwür" charakterisierte; oder die Angriffe von Senior-Vice-President Craig Mundie, der im selben Jahr die Open-Source-Praxis für „ungesund" und zum „Sicherheitsrisiko" erklärte.[7] Vor allem Ratlosigkeit sprach aus den Attacken. Nicht nur hat sich die ungeplant, in globaler Kooperation produzierte Software allzu oft derjenigen qualitativ überlegen erwiesen, die kommerziell hergestellt wird. Auch greifen die üblichen Methoden der Konkurrenzabwehr kaum, mit denen Konzerne wie *Microsoft* über die Jahrzehnte so erfolgreich waren. Der ungewohnte neue Gegner scheint ebenso allgegenwärtig wie unfassbar.

Open-Source-Projekte stellen daher eindeutig mehr als nur eine weitere Konkurrenz unter den vertrauten anderen dar. Ihr Erfolg beruht keinesfalls nur darauf, dass die in ihrem Kontext hergestellten Werke am Ende gratis oder zumindest vergleichsweise billig abgegeben werden. Von größerer und langfristigerer Bedeutung sind strukturelle Innovationen, die im Kontext der Produktion freier Software gelangen. Sie zeigen

5 Zahlen von Dezember 2000 nach http://survey.netcraft.com/index-200012.html. Zahlen von Dezember 2006: 60,3 Prozent *Apache* vs. 31,04 Prozent *Microsoft*; nach http://news.netcraft.com/archives/web_server_survey.html.

6 Vgl. z. B. http://en.wikipedia.org/wiki/Source_lines_of_code.

7 Zu den *Halloween Papers* vgl. http://www.catb.org/~esr/halloween. Zu Ballmer vgl. z. B. Wilcox und Shankland (2001). Zu Mundie vgl. z. B. Charny (2001).

die Open-Source-Praxis, so lautet die Grundeinsicht dieses Essays, als Teil eines historischen Trends, eines nachhaltigen Wandels der Produktions-, Distributions- und Kommunikationsweisen – als eine treibende Kraft im Übergang von einer Zivilisation, die auf industrieller Technologie beruht, zu einer Zivilisation, die sich digitaler Technologie bedient.

Absicht der folgenden historischen Skizze ist es daher, den Ursprung und die zentrale Funktion der Open-Source-Praxis in diesem epochalen Prozess der Formung und Durchsetzung digitaler Technologie und Kultur zu beschreiben und theoretisch zu fassen. Eine Begriffsbestimmung muss dazu vorweg geschehen: Die Rede von der Open-Source-Praxis bezeichnet hier keineswegs nur Handlungs- und Verfahrensweisen im engeren Zusammenhang der in den neunziger Jahren begründeten Open-Source-Bewegung selbst. Der Begriff zielt vielmehr auf deren soziokulturellen Ursprung – auf innovative Praktiken, die sich seit der Mitte des 20. Jahrhunderts im avancierten Umgang mit digitaler Technik zunächst in akademischen Nischen und jugendlichen Subkulturen ausbildeten, mit dem Fortschreiten der Digitalisierung immer breitere Kreise erfassten und schließlich gegen Ende des Jahrhunderts, als auch der Begriff der Open Source im gegenwärtigen Sinne allererst geprägt wurde, zunehmend ritualisierte, kodifizierte und institutionalisierte Formen gewannen.[8]

Eine vollständige Darstellung dieses mehr als ein halbes Jahrhundert umfassenden Prozesses würde freilich den Rahmen eines Aufsatzes sprengen. Meine Skizze der historischen Entfaltung der Open-Source-Praxis und ihrer grundlegenden Prinzipien konzentriert sich daher auf drei Schwerpunkte. Sie steuern die notwendige Selektion des historischen Materials und betreffen die sukzessive Ausbildung erstens einer neuen Organisation der (Wissens-)Arbeit, zweitens einer neuen Organisation der Verfügung über die Produkte dieser (Wissens-)Arbeit sowie drittens einer neuen, die industrielle Kultur der Massen wenn nicht ablösenden, so doch ergänzenden und modifizierenden digitalen Kultur der Einzelnen.

Meine These zu dem ersten Schwerpunkt lautet: Die Open-Source-Praxis bedeutet für die digitale Epoche, was die Praxis des Taylorismus bzw. Fordismus der industriellen Epoche war – eine strukturbildende Leitidee, die ausgehend von der Arbeitsorganisation soziale Verhaltens- und Denkweisen prägt, ebenso die Organisation des zivilisatorischen Wissens sowie nicht zuletzt auch Kunst und Unterhaltung.

Ermächtigt von digitaler Technologie, hat die Open-Source-Praxis im Bereich der Produktion und Distribution von Programm-Software einen radikalen Wandel vollzogen. Insbesondere mit den Prinzipien freiwilliger Assoziation und Selbstorganisation entwirft sie, wie selbst das eher konservative Wirtschaftsmagazin *Forbes* bereits 1998 erkannte: „a neat alternative way to produce better software" (McHugh 1998). Das allein wäre nicht wenig. Doch die Bedeutung der Neuerungen reicht über den Bereich der Softwareherstellung weit hinaus. Denn traditionell pflegen jene Branchen, die sich der fortgeschrittensten Technologien bedienen und sie fortentwickeln, den

8 Zum Begriff Open Source siehe Abschnitt 10.

anderen neue Wege zu weisen. Die avancierten Industrien jeder Epoche fungieren als Laboratorien. Von Innovationsdruck getrieben erproben sie neue Arbeits- und Verhaltensweisen, Rechtsformen, Organisationsmodelle, Managementverfahren, die Produktivitäts- und Wettbewerbsvorteile bieten und daher früher oder später andere Bereiche der Produktion, ja die ganze Gesellschaft erfassen – womit wiederum neue, den gewandelten Arbeits- und Lebensweisen gemäße Unterhaltungs- und Konsumbedürfnisse entstehen.[9]

In der industriellen Frühzeit spielte etwa die Textilindustrie eine solche Schrittmacherrolle, gegen Ende des 19. Jahrhunderts die Stahlindustrie und seit Beginn des 20. die Autoindustrie. Sie nahmen jeweils vorweg, was später das gesamte Wirtschaftsleben umwälzte; nicht nur die Produktion, sondern auch die Distribution, den Handel, die Verwaltung und die Dienstleistungssektoren. Richtungsweisende Beispiele dafür geben die Taylorisierung bis hin zur Einführung des Fließbands, die von der Autoindustrie ausging, oder aber ab der Mitte des 20. Jahrhunderts die generelle Übernahme industrieller Technologien und Arbeitsprinzipien in die Agrarwirtschaft. In welche Richtung nun gegenwärtig die Überwindung der überkommenen industriellen Ordnung geht, zeigen die IT-Branchen und insbesondere die avantgardistische Open-Source-Praxis.

Kaum freilich ist zu erwarten, dass die industrielle Ordnung komplett durch eine neue digitale ersetzt würde. Historischer Fortschritt bevorzugt Akkumulation. Die industriell-automatisierte Produktionsweise verdrängte die mechanisch-handwerkliche nicht, sie marginalisierte und modifizierte bzw. modernisierte sie. Ähnlich darf man sich den Übergang in die digitale Epoche vorstellen. In deren Zentrum stehen, da die hervorragenden Produkte digitaler Technologie Software sind, Fragen des Eigentums an und der Nutzung von immateriellen Artefakten.

Meine These zu dem zweiten Schwerpunkt der Verfügung über die Produkte der neuen Wissensarbeit lautet: Die Varianten der Open-Source-Regelungen geistigen Eigentums suchen unter den Bedingungen digital vernetzter Wissensproduktion und ihrer primär virtuellen Resultate herzustellen, was die Varianten industrieller Regelungen geistigen Eigentums unter den Bedingungen der Fabrikproduktion und ihrer primär materiellen Resultate leisteten – einen für Individuen und Kollektive sowohl erträglichen wie fruchtbaren Ausgleich der jeweils divergierenden Interessen von Produzenten und Rezipienten, Schöpfern und Nutzern.

Unübersehbar kollidiert das geltende Urheber- und Patentrecht, dessen Prinzipien der Frühzeit der Industrialisierung verpflichtet sind, mit dem Potenzial digitaler Technologien. Der Versuch, die industrielle Ordnung in der digitalen Epoche aufrechtzuerhalten, hemmt in der Praxis nachhaltig die Wissensarbeit und kriminalisiert

9 Diese Vorbildfunktion fortgeschrittener Gewerbe und Branchen ist schon in der philosophischen Reflexion und politischen Theorie von Aufklärung und industrieller Frühzeit erkannt und beschrieben worden, nicht zuletzt von Karl Marx. Die folgende Darstellung stützt sich wesentlich auf Einsichten, wie sie David S. Landes (1969) und Joel Mokyr (1990) entwickelten.

zudem komplette soziale Gruppen beziehungsweise Generationen. An den Modifikationen der veralteten juristischen Rahmenbedingungen, über die gegenwärtig weltweit gestritten wird, hängen wesentlich das Tempo und auch die Richtung der weiteren digitalen Modernisierung.[10]

In keinem anderen Bereich immaterieller Produktion aber ist die Suche nach alternativen Eigentums- und Rechtsformen so weit fortgeschritten wie in der Open–Source-Praxis. Von Richard Stallmans Versuch einer Neuordnung geistigen Eigentums über Linus Torvalds pragmatische Neuorganisation der Arbeit und Eric Raymonds Legitimierung der neuen digitalen Praxis bis zu Lawrence Lessigs Free-Culture-Konzept sind Modelle für den praktischen wie juristischen Umgang mit den Produkten kreativer Arbeit entwickelt worden. Besonders wegweisend sind ihre Ansätze für die Besitzregelung im Falle kollektiver Wissensarbeiten – Computerprogrammen wie Forschungsergebnissen, Spielfilmen wie Spielen –, deren Bestandteile separat nicht verwertbar sind und die unter den gegenwärtig üblichen Bedingungen von einzelnen „Rechteinhabern", juristischen oder natürlichen Personen, als geistiges Kapital vereinnahmt und damit sowohl ihren eigentlichen Schöpfern wie der Öffentlichkeit entzogen werden. Hier ist dem innovativen Verhältnis von individueller und kollektiver Produktion in der durch digitale Produktions- und Kommunikationsmittel ermächtigten Wissensarbeit Rechnung zu tragen. Denn während unter industriellen Bedingungen ökonomischer Erfolg von möglichst vollständiger Konformität des Handelns abhing und abhängt, erfordert digital ermächtigte Wissensarbeit eine möglichst weitgehende Individualisierung.

Die These zu dem dritten Schwerpunkt der Ausbildung einer digitalen Individualkultur lautet insofern: Wie Taylorisierung und Fordismus mit dem Streben nach Standardisierung des Gebrauchs industrieller Mittel und Artefakte sowie der Kollektivierung der Verfügung über sie die zivilisatorischen Konsequenzen aus den Anforderungen industrieller Technologie und der Fabrikarbeit zogen, so zieht die Open–Source-Praxis in ihrer Insistenz auf Personalisierung des Gebrauchs digitaler Mittel und Artefakte sowie der Privatisierung der Verfügung über sie die zivilisatorischen Konsequenzen aus den Anforderungen digitaler Technologie und vernetzter Wissensarbeit.

Den drei zentralen Leistungen der Open-Source-Praxis, wie sie die vorgestellten Thesen behaupteten, soll nun in der Analyse der bisherigen Phasen genauer nachgegangen werden. Dabei werde ich zum Ersten die Entwicklungsschübe schrittweise skizzieren:

---

10 Eine grundsätzliche und leicht verständliche Einführung in die Problematik gibt Vaidhyanathan (2001). Genauere Informationen zur historischen Konstellation zu Beginn des 21. Jahrhunderts finden sich z. B. in Lessig (2004) und Gantz und Rochester (2005). Eine juristisch orientierte Bestandsaufnahme liefert Litman (2006). Eine ebenso aktuelle wie grundsätzliche Darstellung des Zusammenhangs zwischen der gesetzlichen Regelung geistigen Eigentums und der Ausbildung einer neuen digitalen Kultur schließlich verspricht das im kommenden Mai erscheinende Buch von Gillespie (2007).

Erste Phase Historischer Ursprung der Open Source-Praxis (siehe Abschnitt 2).

Zweite Phase Praktische Begründung der Open-Source-Praxis (siehe Abschnitt 4).

Dritte Phase Soziale Durchsetzung der Open-Source-Praxis (siehe Abschnitt 8).

Und zum Zweiten werde ich – in der Reihenfolge ihres Auftretens – die sechs grundlegenden Bestrebungen herausarbeiten, die sich in der Open-Source-Praxis Ausdruck verschafften. Es sind:

Die erste Innovation Der Wille der Nutzer, Souveränität im Umgang mit digitaler Technologie zu erlangen, wie er sich zuerst in den Hacker- und Time-Sharing-Bewegungen der sechziger Jahre zeigte (siehe Abschnitt 3 – Der Wille der Nutzer).

Die zweite Innovation Das Streben der Nutzer nach Kompatibilität und Standards, das sich in der Frühzeit digitaler Vernetzung offenbarte (siehe Abschnitt 5 – Evolution der Standards).

Die dritte Innovation Der Wunsch nach egalitärer Kooperation, wie sie erst entortete dialogische Echtzeit-Kommunikation in Kombination mit dem Prinzip des Peer-Review ermöglicht (siehe Abschnitt 6 – Produktionsgemeinschaft der Gleichen).

Die vierte Innovation Das Verlangen nach geistigem Gemeineigentum, insofern es Informationen über Produkte betrifft, die in Arbeit und Alltag genutzt werden (siehe Abschnitt 7 – Geistiges Gemeineigentum).

Die fünfte Innovation Das Streben nach Selbstorganisation (siehe Abschnitt 9 – Vernetztes Wissensarbeiten).

Die sechste Innovation Die Verschränkung von Konkurrenz und Kollaboration im Interesse der Beschleunigung und Verbesserung kreativer Arbeits- und Entwicklungsprozesse (siehe Abschnitt 10 – Konkurrierende Kollaboration).

Abschließend werde ich dann den aktuellen Stand der Entwicklung und deren möglichen Fortgang bedenken (siehe Abschnitt 11 – Digitale Avantgarde. Resümee und Ausblick).

2 Erste Phase: Historischer Ursprung der Open-Source-Praxis

Einem Bedürfnis vor allem verdankte die Open-Source-Praxis ihre Entstehung: dem Interesse einzelner Anwender – Profis wie Endverbraucher –, begrenzende Fremdbestimmung und bestehende Angebotsmängel zu überwinden. Dabei wurde von Anbeginn Souveränität im Umgang mit Produkten erstrebt, die beruflich oder privat wichtig sind. Welche Art freier Software in der Vergangenheit entstand, hing insofern

stets vom aktuellen Stand der Technik ab und von den zeitgenössischen Varianten unzulänglicher oder unerschwinglicher kommerzieller Produkte oder auch behindernder institutioneller Arrangements. Um diese zu überwinden, bildeten sich in der zweiten Hälfte des 20. Jahrhunderts wechselnde lose Assoziationen. Die ersten – und damit die Anfänge der Produktion freier Software – gehen auf die Zeit der *Mainframes* zurück.[11]

All jene, die damals schon mit Computern arbeiten wollten – Mathematiker, Physiker und Ingenieure, auch die erste Generation professioneller Programmierer und Studenten des neuen Fachs Informatik –, litten bis in die sechziger Jahre des vergangenen Jahrhunderts hinein unter der Unfähigkeit zur interaktiven Nutzung der wenigen *Mainframes* oder Minicomputer. Nicht Einzelne, sondern Kollektive kontrollierten die Computer. Und sie arbeiteten nicht an ihnen, sondern in ihnen. Rechner waren Räume; vollklimatisierte Säle, halb Fabrik, halb Großraumbüro. Ihr Gebrauch orientierte sich streng am industriellen Vorbild: Er verlangte hochgradige Arbeitsteilung in der Bedienung und hierarchisch-bürokratische Verwaltung der knappen Ressource Rechenkraft.

Normalsterbliche, die ihrer Arbeit mit digitaler Rechenkraft aufhelfen wollten, sollten mit den Geräten unter keinen Umständen eigenhändig in Kontakt kommen. Selbst die Mathematiker und Techniker, von denen die Rechner programmiert wurden – der Begriff Software und mit ihm der eigene Beruf des Programmierers kamen erst um 1960 auf –, mussten daher fern der Hardware arbeiten. Ihre Kodierungen speicherten sie auf Stapeln von Lochkarten und trugen sie an die Empfangsschalter der Computerhallen. Ein Verwalter überprüfte dann die Berechtigung des Antragstellers und registrierte die Kartenstapel (*batches*). Irgendwann später einmal wurden sie nach Arbeitsplan technisch eingefüttert (*batch processing*).

Mainframes waren so als unpersönliche Computer konstruiert und wurden zudem so unpersönlich betrieben, wie es nur ging. Gerade die avancierten Nutzer aber, junge Wissenschaftler und Techniker, die das revolutionäre Potenzial des digitalen Computers am deutlichsten erkannten, ersehnten schon Ende der fünfziger Jahre das genaue Gegenteil: persönliche Rechner. Denn erst in der Kombination aus jederzeitiger Verfügbarkeit und freiem interaktiven Umgang konnte die digitale Rechenkraft ihr Potenzial entfalten, die Leistungsfähigkeit einzelner Wissensarbeiter zu steigern. Die industrielle Verwaltung der Rechenkraft dagegen behinderte – die Nutzer wie den digitalen Fortschritt. Die Rebellion gegen hierarchische Zugangsbeschränkungen und geheimniskrämerische Vorschriften ließ so um 1960 die erste Generation von Hackern entstehen. Fast zeitgleich taten sich an mehreren amerikanischen und bald auch britischen Universitäten andere unzufriedene Nutzer zusammen, um terminalzentrierte Time-Sharing-Verfahren zu entwickeln. Sie ermöglichten zum ersten Mal Einzelnen den direkten und auch spielerischen Kontakt mit Computern.

11 Die historische Darstellung folgt Levy (1984); Campbell-Kelly und Aspray (1996); Ceruzzi (1998); Abbate (1999); Friedewald (1999); Waldrop (2001).

Die Radikalität des Wandels, der damals begann, zeigt sich an nichts deutlicher als an *SpaceWar!*. Das erste aller Computerspiele schrieben MIT-Studenten 1962. Die freie Software breitete sich wie ein Lauffeuer über die universitären Computer aus. Mit ihrer demonstrativen Verschwendung teurer Rechenzeit ersetzten die jugendlichen Hacker, wie Allucquere Rosanne Stone schrieb, im Umgang mit Computern die Arbeitsethik durch eine Spielethik, das ökonomische Effizienzprinzip der kollektiven Organisationen durch das luxurierende Lustprinzip des Individuums (Stone 1995, S. 13 ff.).

Anwender wurden so, da sie mit dem, was ihnen angeboten wurde, unzufrieden waren, zu Entwicklern. Sowohl das Bedürfnis nach einem interaktiven und selbst bestimmten Umgang mit den neuen Rechenmaschinen als auch die Entschlossenheit, solchen Umgang zur Not gegen die Autoritäten durchzusetzen, welche die Rechenzeit nach dem Vorbild industrieller Fabriken verwalteten, indizierten einen einschneidenden Mentalitätswandel: Das industrielle Verständnis von Technik und ihren zivilisatorischen Aufgaben, wie es aus der Dampfmaschine oder dem System der Elektrizitätsversorgung resultierte, begann mit einer Auffassung zu kollidieren, die bereits vom Umgang mit elektronischer und digitaler Technik instruiert war.

Diese neue Haltung verhielt sich zu der industriellen konträr. Denn der Arbeiter, den Taylorismus und Fordismus zu schaffen suchten, war als Rädchen in der industriellen Maschinerie gedacht. Abläufe pflegten durch vereinfachende Zerlegung auf ein standardisiertes Niveau gebracht zu werden, welches die reibungslose Integration des Arbeitenden und seiner Handgriffe in die Apparatur erlaubte. Industriellen Maschinen wie dem Fließband war gewissermaßen das Kollektiv zu seiner Bedienung eingebaut.[12] Digitale Computer hingegen, wie sie John von Neumann konzipierte, sind dem Gehirn nachgebildet. Sie helfen bei Wissensarbeiten – kalkulieren, kommunizieren, kreieren. Ihnen bzw. ihrer Software ist daher gewissermaßen der Einzelne eingebaut, der sie nicht einfach nach Vorschrift bedient, sondern mit ihnen dialogisch, interaktiv umgeht. Solange sie also als digitale analoge Maschinen behandelt wurden, als Gerätschaften, die vorgegebenen Zwecken dienten und mit menschlicher Hilfe vorgeplante Abläufe exekutierten, setzte man sie entgegen ihren besonderen technologischen Qualitäten ein.

Produktiver als der nach industrieller Gewohnheit abgerichtete Umgang war das Zusammenwirken, ein relativ offenes Wechselspiel von Mensch und Maschine, wie es in den frühen sechziger Jahren gleichermaßen die Avantgardisten der Computertheorie[13] und die erste Generation der Hacker zu propagieren begannen: informelle und egalitäre Selbstverwaltung. Verteilt werden sollte die knappe Rechenzeit nicht länger nach institutionellem Rang, sondern nach Talent und kreativer Leistung. Dieser Offenheit des Zugangs sollte die Offenlegung der Arbeitsergebnisse entsprechen. Auf Mainframe- und Minicomputern, die stets von mehreren genutzt wurden, die Daten

12 Vgl. Abschnitt 9 zum Verhältnis von Taylorismus und Open-Source-Praxis.

13 Vgl. insbesondere Licklider (1960) und Engelbart (1962).

eines Nutzers vor den anderen zu verbergen, kostete einige Anstrengung und erschien den Hackern unnatürlich, ein überflüssiges Hindernis. Geheimnisse bremsten zudem den technischen Fortschritt. Wozu man keinen Zugang hatte – etwa den Quellcode von Programmen –, das konnte man nicht verbessern. Die bis in die Gegenwart einflussreiche Grundregel der Hacker-Ethik, die in den sechziger Jahren am *MIT* formuliert wurde, lautete folgerichtig: *All information should be free.*[14]

Die Erfahrungen, die damals von der Generation früher Anwender gewonnen wurden – mit ersten Computerspielen wie *SpaceWar!*, mit Time-Sharing-Software, mit einfachen Programmiersprachen wie *Basic* –, bedeuten insofern nicht nur generell den Anfang populärer Computerkultur. Der avantgardistische Aufstand dieser kleinen, wenn auch professionellen Gruppen von Nutzern, zu dem es in den frühen sechziger Jahren kam, markiert darüber hinaus ein gutes Vierteljahrhundert bevor der Begriff zuallererst geprägt wurde den Ursprung der Open-Source-Praxis.

3 Die erste Innovation: Selbstermächtigung der Nutzer

In dieser Frühphase der Digitalisierung bereits schälte sich damit die vielleicht wichtigste Lehre heraus, die sich aus dem gewandelten Umgang mit avancierter Technik ziehen lässt: Nichts beschleunigt praktischen und dann auch ökonomischen Erfolg so sehr wie die Einbeziehung derjenigen, die mit den jeweiligen Produkten arbeiten und leben müssen. Umgekehrt jedoch behindern verhärtete Gleichgültigkeit von Großinstitutionen und kurzsichtiger Eigennutz von Firmen einen Fortschritt, der gerade diesen Institutionen und Firmen beachtliche Vorteile bringt, wenn er nur von engagierten Nutzern erst einmal durchgesetzt worden ist (siehe von Hippel 2005). Die erste Bestrebung der Open-Source-Praxis besteht daher in der (Selbst-)Ermächtigung der Nutzer – derjenigen also, denen zu industriellen Zeiten als Kunden oder Rezipienten, professionellen Anwendern, Hobbyisten oder Endverbrauchern eher der gehorsame Nachvollzug des technischen Angebots zugedacht war.

Dieser Rollenwandel parallel zur Durchsetzung digitaler Technologie lässt derweil erkennen, was die industrielle Praxis und die von ihr geprägte Mentalität lange verbargen. „The normal literature focuses on innovation development purely as something manufacturers do", sagt Eric von Hippel, Ökonom an der *Sloan School of Management* des *MIT*. Von Hippels Forschungen ergaben, dass Verbraucher und Anwender schon immer für einen wesentlichen Teil aller Neuerungen verantwortlich waren: „We have shown that users develop many or most of the important innovations in fields ranging from semiconductor process equipment to scientific instruments, and in many consumer goods, ranging from cakes and cookies to garments and sports equipment as well." Instruktive Beispiele bieten Müsliriegel oder Skateboards (von Hippel 2000).

14 Vgl. Levy (1984, S. 40). Bekannter und gebräuchlicher ist heute die Schlachtruf-Formulierung, die Stewart Brand in dem Jahr prägte, in dem auch Levys Buch erschien: „*All information wants to be free.*" (http://www.anu.edu.au/people/Roger.Clarke/II/IWtbF.html)

Unter analogen Verhältnissen freilich bestand – wollten Verbraucher ihre eigenen Interessen durchsetzen – nur die Wahl zwischen zwei Extremen. Einzelne konnten in individueller Heimarbeit ein bestehendes Produkt verbessern oder neu konstruieren. Oder sie konnten versuchen, gegen den institutionellen Widerstand beziehungsweise die Gleichgültigkeit der Produzenten indirekten Einfluss zu nehmen, etwa durch das Einreichen von Verbesserungsvorschlägen oder durch Teilnahme an Marktuntersuchungen. Solches Kundenfeedback mochte zu partiellen Re-Designs führen. Erst aber die Durchsetzung digitaler Vernetzung ließ die Einbeziehung von Anwendern eine funktionale Qualität gewinnen und Nutzer unabhängig von ihrem Aufenthaltsort zu potenziell gleichberechtigten Mit-Entwicklern und Mit-Arbeitern werden.

Von daher erscheint es nur zwangsläufig, dass auch die Basisinnovationen digitaler Vernetzung – von der Paketschaltung (*packet switching*) über die E-Mail bis zu den Transportprotokollen des Internets – primär außerhalb des kommerziellen Sektors gelangen, als eben Teil der Selbstermächtigung von Anwendern zu jenen Handlungen und Verfahrensweisen, die wir heute unter dem Begriff Open-Source-Praxis fassen. Deren zweite Phase – die Gründerjahre, wie generell die digitaler Kultur – begann, als in den siebziger und achtziger Jahren die technischen Voraussetzungen für eine interaktive und entortete Kollaboration in Echtzeit geschaffen wurden.

4 Zweite Phase: Praktische Begründung der Open-Source-Praxis

Um 1970 kulminierten Entwicklungen in drei parallelen Strängen der Digitalisierung. Binnen weniger Jahre wurde so die technologische wie soziale Basis der Open–Source-Praxis gelegt – nicht freilich von Institutionen und Konzernen, sondern durch die Initiativen einzelner Anwender. Zum Ersten ließ das Bedürfnis nach individueller Verfügung über Computer engagierte Bastler im offenen Austausch untereinander die für Taschenrechner 1969 entwickelten Mikroprozessoren zweckentfremden. Diese hobbyistischen Hardware-Hacks begründeten im zweiten Drittel der siebziger Jahre die PC-Industrie, deren Siegeszug Anfang der achtziger Jahre dann die erste Organisation der Open-Source-Praxis erzeugen sollte. Zum Zweiten nahm sich 1969 der Programmierer Ken Thompson eines Projekts an, das *Bell Labs*, *MIT* und *General Electric* nach hohen Investitionen schlicht aufgegeben hatten: Aus dem Mehrplatzbetriebssystem *Multics* entwickelte er zusammen mit Dennis Ritchie, der zur gleichen Zeit die Programmiersprache *C* schuf, ein Einplatzsystem. Folgerichtig wurde es *UNIX* getauft. Es sollte für die nächsten dreißig Jahre die avancierte Software-Entwicklung für Mikrocomputer, PC und Internet prägen.

Zum Dritten schließlich mündete der jahrelange Trend zum vernetzten Echtzeitrechnen 1969 in die ersten Knoten des *Arpanet*. Aus ihm entstand ein Jahrzehnt später dank des offenen TCP/IP-Protokolls das Internet und damit das entscheidende Produktionsmittel für freie Software. Die sukzessive Eröffnung des Datenraums

seit den siebziger Jahren mit der Möglichkeit zum Transfer von Daten, mit E-Mail und Usenet-Foren versetzte dem keimenden Trend zur freien Softwareentwicklung zunächst einen qualitativen, später auch einen quantitativen Schub. Die digitalen Netze verbanden die geografisch verstreuten und voneinander isolierten Widerstandsnester, die sich an Institutionen wie *Bell Labs*, *MIT* oder der *University of California* in Berkeley gebildet hatten. Rund um die Produktion freier Software formte sich eine erst nationale und seit den achtziger Jahren zunehmend internationale Gemeinschaft.

Angesichts der damals recht rudimentären Gestalt globaler Vernetzung stand im Vordergrund der Bemühungen zunächst der nutzergerechte Ausbau des Datenraums selbst. Freie Software entwickelte so wesentlich die Infrastruktur des Internets. Eric Allman von der *University of California* in Berkeley schrieb 1981 *Sendmail* für *UNIX*, um den Austausch elektronischer Post zwischen dem Universitätsnetz, dem er angeschlossen war, und dem weitläufigeren *Arpanet* zu ermöglichen. Ebenfalls Anfang der achtziger Jahre entstand *BIND (Berkeley Internet Name Domain)* – das Protokoll, das bis heute die Eingabe simpler Namen statt jener Zahlenkolonnen erlaubt, aus denen die Netzadressen realiter bestehen. Weitere Beispiele für den Erfolg freier Software im Umfeld des Internets geben *Perl* (*Practical Extraction Report Language*), *Python* oder *PHP*. *Perl*, 1987 von dem Linguisten und damaligen Staatsangestellten Larry Wall entwickelt und zur freien Nutzung im *Usenet* veröffentlicht, war Anfang der neunziger Jahre die Standardsprache zum Entwickeln dynamischer Webinhalte, ohne die E-Commerce kaum denkbar gewesen wäre. Mittlerweile haben zwei andere freie Software-Entwicklungen diese Funktion weitgehend übernommen: *Python*, Anfang der neunziger Jahre von Guido van Rossum entwickelt, und *PHP* (ursprünglich *Personal Home Page*, heute verstanden als Abkürzung für *PHP Hypertext Preprocessor*), entstanden Mitte der neunziger Jahre im Kontext von David J. Hughes' mSQL, einer ursprünglich freien relationalen UNIX-Datenbank.

Freie Software entwickelte so wesentlich die Infrastruktur des heutigen Internets. In dieser engen Verknüpfung zeigt sich entortete Echtzeitkommunikation als Conditio sine qua non der Open-Source-Praxis. Gleichzeitig ließen Leistungssteigerung und Verbilligung PCs in vielen Bereichen an die Stelle der teureren Mikrocomputer treten. Damit entstand das Bedürfnis, freie Software auch für persönliche Computer zu produzieren.

Die Jahre um 1969/70 schufen insofern mit der Einführung des Mikroprozessors bzw. Konstruktion des PCs und der Etablierung der ersten nationalen Netzwerkknoten des *Arpanet* die Voraussetzungen für die in der Folge eskalierende Selbstermächtigung der Anwender, für die Etablierung von offenen Software-Standards und vor allem für die Ausbildung einer historisch gänzlich neuen, der Tendenz nach globalen Produktionsgemeinschaft der Gleichen. Das erste Beispiel für das Potenzial nonproprietärer Software gab *UNIX*.

5 Die zweite Innovation: Evolution der Standards

Die erste UNIX-Version wurde 1975 auf einem Magnetband vertrieben – ohne jede institutionelle Unterstützung, ohne Marketing, ohne *helpline*. Binnen 15 Jahren sollte das Programm all die Mikrocomputer-Betriebssysteme verdrängen, die um 1970 noch den Markt unter sich aufteilten. *UNIX* gab so – vor allem zwischen 1969 und 1979, in der Zeit, die auch den PC hervorbrachte – ein frühes und ungemein erfolgreiches Beispiel für die Effektivität freier Software. Der Erfolg ließ bald auch proprietäre UNIX-Versionen auf den Markt kommen. Bereits *UNIX*' Version 7 sowie *Xenix*, die erste – partiell der damaligen Kleinfirma *Microsoft* zu verdankende – Portierung von *UNIX* auf einen Intel-Prozessor, verschlossen 1979 den Quellcode selbst für akademische Zwecke. Langfristige Nachwirkungen zeitigte der Siebziger-Jahre-Erfolg von *UNIX* noch in manch anderer Hinsicht. Die von dem Studenten Bill Joy 1977 in Berkeley entwickelte Variante etwa stellte 1982 die Software-Grundlage der heutigen Milliardenfirma *SUN*. Die Leistungsfähigkeit von *UNIX* inspirierte ebenso das von Richard Stallman 1984 begründete freie GNU-Projekt, mit dem ein freies, UNIX–artiges Betriebssystem geschaffen werden sollte (*GNU* ist ein rekursives Akronym für *GNU's Not UNIX*). Die vorhandenen Elemente der GNU-Variante wiederum halfen 1991 Linus Torvalds, den Linux-Kernel zu programmieren und zu kompilieren. Die GNU-Bewegung inspirierte Torvalds ebenfalls, seinen Kernel als freie Software unter die *GPL (General Public License)* zu stellen. Darüber hinaus schuf das GNU-Projekt die Systemtools und Anwendungen, die Linux erst zu einem benutzbaren Betriebssystem machen – weshalb viele statt der einfachen Bezeichnung Linux den Terminus GNU/Linux bevorzugen. Und *UNIX* selbst, in der Berkeley-Variante, liegt schließlich seit Ende der neunziger Jahre dem Apple-Betriebssystem *Mac OS X* zugrunde, dessen X zugleich die Version 10 wie den UNIX-Ursprung indiziert.

In den siebziger Jahren machte *UNIX* so – wie später viele andere offene Protokolle und Standards – erstmals einen wichtigen Unterschied zwischen freier und proprietärer Software deutlich: Solange Firmen und Institutionen eifersüchtig den Entwicklungsgang bestimmen, pflegen hinderliche Inkompatibilitäten zu wuchern. Lange Zeit war es etwa so gut wie unmöglich, Daten zwischen den Computern verschiedener Hersteller auszutauschen – nicht einmal reine Textdokumente. Bereits Ende der fünfziger Jahre war daher der *American Standard Code for Information Interchange* (*ASCII*) entwickelt worden, der eine solche minimale Interoperabilität gewährleisten sollte. Doch es dauerte fast eine Dekade, bis er akzeptiert, und weitere fünfzehn Jahre, bis er allen Ernstes implementiert wurde und auch nur die nötigsten Sonderzeichen umfasste. Denn mit einer gewissen Zwangsläufigkeit behindern proprietäre Programme und die mit ihnen verbundene Geheimniskrämerei die Ausbildung und Durchsetzung von offenen Standards und damit eine optimale Nutzung.

Ein Grund für den sofortigen Erfolg von UNIX lag daher – bis 1979 zumindest – in der Offenheit des Quellcodes: Sie gewährleistete weitgehende Portabilität,

die Möglichkeit also, das Betriebssystem auf Mikrocomputern verschiedener Hersteller laufen zu lassen. Das war eine praktische Neuerung, die bezeichnenderweise durch Anwender ausgelöst wurde. Die damaligen Firmen nämlich setzten ihre proprietären Betriebssysteme als Mittel im Konkurrenzkampf ein. Deren Einzigartigkeit ging so weit, dass sich oft nicht einmal Daten zwischen verschiedenen Modellen derselben Hersteller austauschen ließen und sie beim Kauf eines anderen Computers – gleichsam wie beim Übertrag von einem analogen Speicher in einen anderen – von Hand neu eingegeben werden mussten. *UNIX*, von Nutzern geschaffen und zunächst ohne Firmenunterstützung popularisiert, beendete diese widersinnige Beschränkung des Potenzials digitaler Technologie und konnte nicht zuletzt deshalb zügig andere Mikrocomputer-Betriebssysteme vom Markt drängen.

In der Entwicklung freier Software, das lehrt das frühe Beispiel, das *UNIX* gab, setzen sich Standards, welche die Arbeit und den Austausch erleichtern, zügiger, weil naturwüchsig durch. Dafür gibt es subjektive wie objektive Gründe. Aus der Wirtschaftstheorie ist bekannt, dass Marktmechanismen weit über Ab- und Einsichten von Individuen wie einzelnen Gruppen hinaus geordnete Systeme zu initiieren vermögen – weshalb am Ende nahezu aller Entwicklungszyklen Standards stehen. Im Falle der Closed-Source-Verhältnisse beschränkt sich die Wirksamkeit solcher Selektionsmechanismen weitgehend auf komplette Produkte. Open-Source-Strukturen hingegen dehnen die Wirkung der Marktmechanismen bereits auf den Entwicklungsprozess aus, wodurch der Standardisierungsdruck eskaliert. Denn Anwender sind, egal, wieviel sie für eine Ware bezahlt haben, primär an deren Gebrauchswert interessiert. Sie opponieren daher allem, was eine effektive Nutzung beschränkt – vor allem Inkompatibilitäten mit anderen Hard- und Softwareprodukten, die sie oder Menschen in ihrem Umfeld bereits einsetzen.

Dieser Ausdehnung von Marktmechanismen auf den Entwicklungsprozess verdankt sich die notorische, den gesunden Menschenverstand verblüffende Einheitlichkeit und historische Kontinuität der Open-Source-Projekte. Man sollte erwarten – und selbst die meisten Entwickler in der freien Softwarebewegung taten das anfänglich –, dass die ungeplante und weitgehend unkontrollierte Arbeit von Hunderten oder Tausenden von Personen, die über den Erdball verteilt leben und sich kaum je *face-to-face* treffen, zur inhaltlichen Zersplitterung, zur Kurzlebigkeit von Projekten und damit zu einem höheren Maß an divergierenden Entwicklungen und Inkompatibilitäten führt. All dies geschah zwar in der Geschichte freier Software. Die drei freien BSD-Varianten *FreeBSD*, *NetBSD* und *OpenBSD* etwa zeugen von der immer möglichen Zersplitterung. Und natürlich kamen viele Open-Source-Projekte – nicht anders als in der Closed-Source-Welt – kaum über ihre Anfänge hinaus und gingen sang- und klanglos unter.

Doch diesen durchaus zahlreichen, wenn auch schnell vergessenen Ausnahmen steht die beachtliche Stabilität vieler großer Projekte gegenüber. Sie resultiert aus der spezifischen Struktur des Open-Source-Prozesses. In ihm nämlich konkurrieren alle

Projekte um eine begrenzte Zahl von talentierten Entwicklern. In der Konsequenz führt das zu einer beschleunigten Selektion: Unattraktive Projekte verlieren recht schnell ihre besten Mitarbeiter zu Gunsten populärerer Projekte, die sich dadurch zügig stabilisieren. Und selbst in den Fällen, da zwei oder gar mehr konkurrierende Ansätze überleben, ist unbeabsichtigte Inkompatibilität allein aufgrund des offenen Quellcodes so gut wie ausgeschlossen. Einerseits also kann Unverträglichkeit kaum zufällig entstehen, andererseits macht im Gegensatz zur Closed-Source-Welt auch ihre gezielte Produktion wenig Sinn, da sich mit mangelnder Kompatibilität bei gleichzeitig offenem Quellcode keinerlei Konkurrenzvorteile, vielmehr sogar Nachteile im Hinblick auf die Akzeptanz verbinden.

Aus diesen strukturellen Gründen neigt kollektives Design im historischen Verlauf statt zur Kurzlebigkeit zu Kontinuität und statt zu Diversifikation zur Standardisierung.[15] Daraus ergibt sich der Gesamteindruck des Gegensatzes zum hierarchisch verwalteten Closed-Source-Bereich. Dessen primärer Anreiz zur Mitarbeit besteht in der finanziellen Entlohnung, und dementsprechend bestimmt Fluktuation das Bild: Programmierer finden besser bezahlte Stellungen und verlassen ein Projekt, zahlreiche Firmen verschwinden durch Übernahme oder Schließung vom Markt, die meisten Produkte haben eine kurze Halbwertzeit. In den diversen Open-Source-Nischen, deren Mitglieder um den besten Code wetteifern, herrscht dagegen relative Dauerhaftigkeit der wichtigsten Projekte, personelle Konstanz in den Entwicklungsteams sowie eine langfristige Kompatibilität der Produkte vor.

Das älteste, bis in die digitale Vorzeit der sechziger Jahre zurück reichende Beispiel gibt dafür die dreißigjährige Geschichte der UNIX-Familie in ihrer charakteristischen Mischung aus Generationen-Zersplitterung zwischen proprietären und offenen Varianten bei grundsätzlicher familiärer Kontinuität. Wirklich bedeutend für die Entwicklung freier Software und die damit verbundene Ausbildung von Standards war *UNIX* zwar allein in seinem ersten Jahrzehnt, den siebziger Jahren. Doch sein freies Erbe wirkt fort, nicht zuletzt in der – wesentlich auf *UNIX* basierenden – Infrastruktur des Internets mit seinen offenen Protokollen wie *TCP/IP*, *SMTP* oder *HTTP*.

6 Die dritte Innovation: Produktionsgemeinschaft der Gleichen

Einen weiteren Schwerpunkt des Wandels, den digitale Vernetzung seit den siebziger Jahren auslöste, bildete die Aufrüstung und Umfunktionierung des akademischen Verfahrens einer offenen Begutachtung von Forschungsergebnissen durch die Gemeinschaft der Forscher, des so genannten Peer-Review. Keine andere Praxis prägte über die Jahrhunderte hinweg die Produktion von Wissen so nachhaltig. Denn der „organized scepticism“, wie Robert K. Merton im Rahmen seiner CUDOS-Regeln den aus der Antike überkommenen kritischen Dialog der Gebildeten nannte, garantiert die

15 Hinzu traten in den neunziger Jahren nachhaltige Modifikationen in der Organisation der Arbeit selbst und damit in der Motivation der Arbeitenden. Vgl. dazu Abschnitt 9.

Qualität auf dem Markt des Wissens.[16] Mit der sukzessiven Verwissenschaftlichung von Wirtschaft und Alltag im Prozess der Industrialisierung gewann daher die Expertenkritik auch außerhalb des engeren Bereichs von Wissenschaft und Forschung an Bedeutung. Zu analogen Zeiten begrenzten den Peer-Review jedoch dreierlei Mängel.

Der erste war kommunikationstechnischer Art: Peer-Reviews unterlagen, solange sie an die Publikation und Rezeption in Büchern und Zeitschriften gebunden waren, deren langen, nach Monaten und Jahren rechnenden Zyklen. „[T]he process was achingly slow and cumbersome. In academia, for example, scholars exchanged views via journals, with peer rebuttals or supportive commentary appearing months after a new piece was published" (Tapscott und Ticoll 2000). Das Verfahren musste, solange Interaktion in Echtzeit ausgeschlossen war, ein Mittel der Post-festum-Kritik bleiben.

Der zweite Mangel war primär wirtschaftlich-juristischer Art: Copyrights und Patente wie vor allem die – nicht nur außerhalb des wissenschaftlichen Diskurses verbreitete – Praxis, Forschungsergebnisse und (Firmen-)Wissen im Interesse zukünftiger Profite nicht mit Außenstehenden oder gar Konkurrenten zu teilen, limitierten drastisch die Möglichkeiten kreativen Austauschs. Ohne freien Informationsfluss, ohne Kommunität der Erkenntnisse konnten Peer-Reviews nicht wirklich ihre produktive Kraft entfalten.

Der dritte Mangel war primär sozialer Natur: Er betraf den Zugang zum Kreis derjenigen, die an der Expertenkritik partizipieren konnten. Eingeschränkt wurde er sowohl geografisch, denn Entfernung von den Zentren der Forschung war bis ins letzte Drittel des 20. Jahrhunderts hinein schwerlich zu kompensieren, wie vor allem durch medialen Mangel: Analoge Publikationsmedien haben samt und sonders ihre Beiträger zu selektieren. Mit dieser – wesentlich aus der Materialität der Medien rührenden – Beschränkung war nicht nur die Beteiligung einer möglichst großen Zahl von Experten, sondern auch die normaler Nutzer statistisch ausgeschlossen, derjenigen also, die in ihrer Arbeit oder im Alltag mit den betreffenden Verfahren und Produkten Erfahrungen gesammelt hatten, welche sie zum Peer-Review für andere befähigten, die gleichen Interessen folgten und ähnliche Probleme lösen wollten.

Diese drei grundsätzlichen Mängel, denen die Praxis des Peer-Review unter den Bedingungen analoger Kommunikation unterlag, überwand die Open-Source-Praxis sukzessive. Den Anfang machte seit den siebziger Jahren die Beseitigung der kommunikations- und medientechnischen Mängel: Mit der Verlegung in den Datenraum beschleunigte sich der entortete Austausch von Kommentar und Korrektur potenziell auf Echtzeit. Damit veränderte sich sein Charakter so radikal, wie sich etwa transkontinentale Kommunikation verändert hatte, als sie statt in Briefen, die Wochen mit der Schiffspost unterwegs waren, über Telefon oder im Chat erfolgen konnte: Was eine Serie monologischer Post-festum-Beurteilungen gewesen war, wandelte sich, sobald die Begutachtung von Verfahren und inhaltlichen Vorschlägen ohne größeren Zeit-

16 Siehe Merton (1973). *CUDOS* steht für *Communalism*, *Universality*, *Disinterestedness*, *Originality*, *Skepticism*.

verlust geschehen konnte, zu dialogischer Interaktion. Das überkommene Medium der Kritik für geografisch zerstreute *peer groups* mutierte damit zur Technik selbst organisierter Kollaboration und Kreation. Ein frühes Beispiel dafür gab die Evolution digitaler Vernetzung durch das – 1969 zunächst noch in Papierform begonnene – Request-for-Comments-Verfahren.

Die Aufhebung der medientechnischen Beschränkung durch digitale Echtzeitvernetzung schuf zugleich auch die Voraussetzung für die Ausbildung virtueller Gemeinschaften und damit für die Reduktion der medientechnisch induzierten Begrenzungen.

> „The Internet is a very cheap medium for small volume publishers. It is immediate and unlimited in capacity. Search engines can quickly help readers find what they need no matter how many journals there are. And a Slashdot-like peer review process—a process of first publish and then review—is both quicker and probably leads to greater insight." (Cringely 2002)

Mit der Abkopplung von Medien, die Zugangsschranken errichten, demokratisierte sich der Peer-Review: Von einem Instrument der Erkenntnistechnik wissenschaftlicher Eliten wurde er zu einem Mittel der Kritik und Wissensgewinnung für Normalbürger. Der exponentielle Zuwachs an Internetnutzern in den achtziger und vor allem neunziger Jahren brachte zudem die kritische Menschenmasse für das, was Soziologen den Delphi-Effekt nennen: den Umstand, dass die Zuverlässigkeit von Expertenentscheidungen mit der Zahl der Beteiligten steigt (Raymond 1999, S. 31).

Derweil sind Zusammenschlüsse von Experten und Anwendern zu nahezu jedem denkbaren Interesse lebensfähig – und in ihnen kollaborative Produktionen praktikabel. Die Situation zu Beginn des 21. Jahrhunderts hat Howard Rheingold beschrieben:

> „Note the rise of Web sites like eBay (auctions), Epinions (consumer advice), Amazon (books, CDs, electronics), Slashdot (publishing and conversation) built around the contributions of millions of customers, enhanced by reputation systems that police the quality of the content and transactions exchanged through the sites. In each of these businesses, the consumers are also the producers of what they consume, the value of the market increases as more people use it, and the aggregate opinions of the users provide the measure of trust necessary for transactions and markets to flourish in cyberspace." (Rheingold 2002, S. XIX)

Leser von Online-Magazinen kommentieren Artikel, ergänzen oder korrigieren sie. Rezipienten von Büchern und Videospielen, Filmen oder Musikaufnahmen, ebenso die nichtprofessionellen Nutzer digitaler Hard- und Software sowie anderer Konsumgüter veröffentlichen, was ihnen an den Produkten ge- und missfiel. Längst übertreffen die Millionen kritischer Erfahrungsberichte, Warnungen und Empfehlungen, die sich im Datenraum sammeln, an praktischem Nutzwert wie sozialem Einfluss auf Kauf-

und Rezeptionsentscheidungen alle älteren Verfahren der Kritik, von der journalistischen Rezension über die Resultate der Warentests bis zu den Ratschlägen gedruckter Ratgeber oder kommerzieller Hotlines. Ebenso finden heute in Foren, die sich mit Hard- oder Softwareproblemen bestimmter Produkte beschäftigen, Nutzer aus aller Welt binnen kürzester Zeit Lösungen, die weder einzelne Verbraucher noch die betreffenden Firmen so schnell hätten beibringen können.

Mit dem *WWW* demokratisierte sich daher die Verfügung über die Technik des Peer-Review. Ihr erschwinglicher und unkontrollierter Gebrauch nach partikularen Interessen wurde möglich. Neben den Leistungen digital ermächtigter Amateure erlaubt der globale Talentpool an freiwilligen Beiträgern, im Bereich von Forschung und professioneller Produktion qualitativ hochwertige Ergebnisse zu erzielen. Denn der Zugang zu dem gewaltigen Wissensreservoir, welches das *WWW* derweil birgt, machte – in Verbindung mit der Verbesserung digitaler Simulationstechnik, die es ermöglicht, aufwendige Experimente virtuell durchzuführen – seit den neunziger Jahren an kleineren Universitäten und selbst in kleineren Firmen grundlegende Arbeiten praktikabel, wie sie noch in den achtziger Jahren nur in Zentren der Forschung oder in den Labors großer Konzerne geleistet werden konnten.[17]

Programmierer und Techniker, Hobbyisten und Normalverbraucher, wo sie leben und für welche Firmen sie arbeiten mögen, können so ihre eigenen dringendsten technischen Probleme beziehungsweise die ihrer Arbeitgeber lösen. Gleichzeitig finden sie soziale Anerkennung durch Gleichgesinnte und Gleichqualifizierte, die ihre Leistungen angemessen zu würdigen verstehen – werden also für den unbezahlten Arbeitseinsatz in der aus den *scientific communities* vertrauten Währung der *peer recognition* belohnt. Tapscott und Ticoll (2000) resümieren den durchschlagenden Erfolg der digitalen Vernetzung für die Open-Source-Praxis: „[It] put the collaborative groupthink process on steroids."

Aus dem Randphänomen, das echtzeitig-interaktiver Peer-Review in den siebziger und frühen achtziger Jahren noch war, wurde so mit der Popularisierung des *WWW* am Ende des 20. Jahrhunderts ein Massenphänomen von nicht zu unterschätzender wissenschaftlicher, wirtschaftlicher und alltagspraktischer Bedeutung – eine zentrale Kulturtechnik für die sich demokratisch und in partizipatorischer Selbstselektion organisierende globale Wissensgesellschaft.

7 Die vierte Innovation: Geistiges Gemeineigentum

Mit dem zunehmenden Erfolg freier Softwareproduktion stellte sich in den frühen achtziger Jahren die Notwendigkeit, sie intellektuell zu begreifen und ihre Arbeitsergebnisse durch neuartige Formen sozialer Regeln wie juristischer Verträge abzusichern. Den ersten Versuch, zu analysieren und kodifizieren, was sich als Programmier- und Distributionspraxis naturwüchsig herausgeschält hatte, unternahm Richard Stallman

17 Vgl. z. B. Cringely (2002).

vom *Artificial Intelligence Labor* des *MIT*. Seinen Ruf in der Szene hatte er sich seit 1975 mit *Emacs* (*Editor MACroS*) erworben . In diversen Varianten ist das Programm bis heute einer der besten Texteditoren und ein unentbehrliches Werkzeug aller Hacker. 1984 begründete Stallman das GNU-Projekt und kurz darauf die *Free Software Foundation*. Ihr vorrangigstes Ziel war (und ist) es, die Freiheit jener Software zu gewährleisten, die im Kontext der Open-Source-Praxis entstand – worunter primär die Freiheit der Nutzer zu verstehen ist, Programme zu kopieren, weiterzuverteilen und zu verändern. Mit seinen einflussreichen Non-Profit-Initiativen sorgte Stallman dafür, dass freie Software auch in den Zeiten von *Microsofts* erfolgreichem Streben nach Desktop-Dominanz lebensfähig blieb. Belohnt und geehrt wurde Stallmans Engagement 1990 mit dem siebenstellig dotierten *MacArthur Genius Grant*.

Dass allerdings Anhänger freier Software Mitte der 1980er Jahre plötzlich eine eigene Institution für notwendig hielten, zeugt nicht nur von den Anfängen sozialen Selbstbewusstseins, sondern auch von einem grundsätzlichen zivilisatorischen Wandel: der damals heraufziehenden – und gegenwärtig eskalierenden – Diskontinuität in der Ordnung des Wissens. Denn jede Zeit erschafft ihren eigenen Raum des Wissens. Wie Wissen produziert und wie das je Bekannte organisiert wird, folgt sozialen Präferenzen. Insofern ist die Geschichte des Wissens zu großen Teilen die seiner Verfügbarmachung. Sie wiederum hat zwei Seiten: erstens den Besitz am Wissen – wer also wie den Zugang zum Wissen kontrolliert; und zweitens die Organisation des Wissens – was sich also wo und wie und wann in Erfahrung bringen lässt. Beides, der Besitz am Wissen wie seine Organisation, verändern sich mit dem technologischen Stand und dessen sozialer Nutzung. Insofern gab es eine agrarisch-ländliche Ordnung des Wissens, eine handwerklich-städtische Ordnung des Wissens, und es gab und gibt eine industriell-großstädtische Ordnung des Wissens.[18]

In ihr, in der von Copyright und Patenten etablierten Eigentumsordnung einerseits und der von Bibliotheken und anderen materiellen Verteilungssystemen geprägten Distributionsordnung andererseits, leben wir noch weitgehend. Gleichzeitig jedoch sind wir Zeitgenossen ihrer Auf- und Ablösung. Verstehen lässt sich, was gegenwärtig geschieht, am besten im historischen Vergleich mit der Phase, in der einst die industrielle Ordnung des Wissens entstand.

Bei dem radikalen Wandel im Umgang mit dem zivilisatorischen Wissensbestand zur Zeit von industrieller und bürgerlicher Revolution ging es zunächst um seine

18 Mit einer – jedenfalls im Rückblick – gewissen Zwangsläufigkeit wurde im zweiten Drittel des 20. Jahrhunderts, in der Phase des Übergangs zur digital ermächtigten Wissensgesellschaft, das Wissen selbst – seine Herkunft, seine kulturelle Strukturierung, seine je zeitgenössischen Gehalte – problematisch und damit für die Geisteswissenschaften thematisch. Michel Foucaults (1971; 1988) bahnbrechende Studien „Ordnung der Dinge" und „Archäologie des Wissens" initiierten, nachdem die selbst disruptiv wirkende Dekonstruktion in den siebziger Jahren zum neuen Dispositv aufgestiegen war, eine Vielfalt der Erkundungen konkurrierender Ordnungen des Wissens. Gemeinsam ist ihnen inmitten der Vielfalt der Differenzen einzig wohl die Einsicht in eine – durch welche Kräfte auch immer – induzierte historische Abfolge der Wissensordnungen. Die obige steht im Kontext jener verbreiteten Anstrengungen, kulturwissenschaftliche Einsichten mit sozial- und technikhistorischen zu verbinden.

Verwandlung in intellektuelles Kapital – also darum, im Interesse des industriellen Fortschritts die Rechte an innovativem geistigen Besitz zu sichern. Denn erst solche juristischen Garantien ermöglichten im Rahmen der ebenfalls neu geschaffenen Rechtsformen industrieller Firmen die private Kollektivfinanzierung von Großkonzepten wie der Errichtung großer Fabriken oder dem Ausbau der Eisenbahn. Eine radikale Modernisierung und institutionelle Absicherung des Patent- und Urheberrechts begann.

In der Frühzeit der Industrialisierung vollendeten sich damit Bestrebungen nach einer tendenziellen Gleichstellung von materiellem und geistigem Besitz. Eingesetzt hatten sie bereits im 17. Jahrhundert als Folge von Mechanisierung und Verbürgerlichung. Unter agrarisch-feudalen Verhältnissen war eine Emanzipation des geistigen Kapitals jedoch von marginalem sozialen Interesse und politisch wie praktisch nur partiell zu realisieren. Erst mit der Industrialisierung erhielt das Streben der bürgerlichen Intelligenz nach patent- und urheberrechtlichem Schutz ihrer Arbeit gesteigerte soziale Dringlichkeit und wurde im Interesse industriellen Fortschritts gelöst. Im 19. Jahrhundert etablierte sich so ein historisch neuer Begriff identifizierbaren geistigen Privateigentums, abgesichert durch entsprechende gesetzgeberische Maßnahmen, neue Institutionen, wie zum Beispiel nationale Patentämter, und auch internationale Abkommen.[19]

Dialektisch zur Verwandlung von Wissen in Kapital im Interesse einzelner Produzenten verhielt sich die Notwendigkeit, dieses Kapital im Interesse der Gesamtwirtschaft in produktive Zirkulation zu bringen. Auch dazu trug die an der Schwelle zur Industrialisierung beginnende Stärkung der Urheberrechte wesentlich bei, da sie zur Veröffentlichung sonst eher geheim gehaltenen Wissens anregte. Das erste moderne Urheberrechtsgesetz, das britische *Statute of Anne*, präsentierte sich dann auch 1709/10 als ein „Act for the Encouragement of Learning".[20]

Neben dem Akt der Veröffentlichung selbst war jedoch der Zugang zu dem veröffentlichten Wissen entscheidend. Da Bücher zur Zeit der bürgerlich-industriellen Revolution das dominierende Speichermedium für zivilisatorisches Wissen waren, gerieten sie ins Zentrum des sozialen Machtkampfes. Die um 1800 staatlicherseits im großen Maßstab teils konfiszierten, teils aufgekauften Bände wurden in der ersten Hälfte des 19. Jahrhunderts zur materiellen Grundlage einer gänzlich neuen industriellen Organisations- und Distributionsform von Informationen, zentriert um öffentliche und neugegründete nationale Bibliotheken.[21] Der Bestand etwa der Pariser *Bibliothèque nationale*, eröffnet 1795, verdoppelte sich durch Zwangsakquisition binnen zwei Jahrzehnten auf über 600 000 Bände. Ähnliches geschah im deutschsprachigen Raum. Notwendig wurde der Bruch mit der etablierten vorindustriellen Eigentumsordnung – die Entprivatisierung, Ver-Öffentlichung und Umorganisierung des

19 Vgl. Fußnote 10, inbesondere Vaidhyanathan (2001).

20 Vgl. z. B. das Facsimile auf http://www.copyrighthistory.com/anne.html.

21 Zur Geschichte der Bibliothek vgl. etwa Battles (2003).

Buchbestandes – im 19. Jahrhundert aus ökonomischen Gründen. Die technologische Aufrüstung der Arbeitsprozesse in den Fabriken und noch mehr der Ausbau nationaler Bürokratien zur Verwaltung der industriellen Massen verlangten nach qualifizierteren Arbeitskräften, als sie unter den Bedingungen agrarisch-feudaler Wissensvermittlung in ausreichender Zahl auszubilden waren. Wissen wurde damit von einem Luxus der Oberschichten zu einem existenziellen Überlebensmittel. Der politische Kampf um Volksbildung, die Umwandlung von individueller Bildung in massenhafte Ausbildung begann.

Die Bibliotheken waren zentrale Agenturen dieses Prozesses. Ihre Philosophie war die akademische. Sie unterschied zwischen materiellen Gegenständen, deren Menge begrenzt war und durch Verbrauch im Wert reduziert wurde, weshalb mit ihnen sparsam umzugehen war, und intellektuellem Kapital, dessen Wert sich durch Gebrauch nicht reduzierte, sondern durch Verbreitung sogar wuchs. Öffentliche Bibliotheken stellten die dem Stand der Reproduktionstechniken angemessene Kompromissform dar: Sie machten das Wissen, das in dem begrenzten Vorrat an gedruckten – und eben seit dem 17. Jahrhundert durch Varianten des Patent- und Urheberrechtschutzes vor unautorisierter Reproduktion geschützten – Büchern gespeichert war, einer maximalen Menge von Menschen zugänglich. Die revolutionäre, mittels juristischer Zwangsmaßnahmen durchgesetzte Ver-Öffentlichung der Buchbestände stellte von daher eine der wichtigsten kulturellen Adaptionsformen an die gewandelten Erfordernisse der Industrialisierung dar.[22]

Vergleichbares vollzieht sich, seit die industrialisierte Welt an die Schwelle zur digitalen Epoche getreten ist. Erneut scheint eine historische Anpassung im Umgang mit dem zivilisatorischen Wissen notwendig. Wissensarbeit verlangt nach globalem und personalisiertem Zugriff auf den zivilisatorischen Bestand. Ihn aber verhindert die industrielle Eigentums- wie Distributionsordnung mehr, als dass sie ihn ermöglicht. Einer der ersten, der aus der Krise der industriellen Wissensorganisation Konsequenzen ziehen wollte, war H. G. Wells. Ende der 1930er Jahre propagierte er die Idee einer „Permanent World Encyclopedia" (Wells 1937). Wells' Ziel war es, die Verwaltung und Verbreitung des Wissens auf den letzten industriellen Stand zu bringen. Das technische Mittel dazu schien ihm die industrielle Technik der Mikrofotografie. Sie ermöglichte, schrieb er, eine fotorealistische Reproduktion, maschinelle Vervielfältigung in hoher Kopienzahl und dank Verkleinerung eine erschwingliche weltweite Distribution aller bis dahin nur an bestimmten Orten zugänglichen Dokumente von Kunst, Kultur und Wissenschaft.

1945 entstand dann die nächste Vision einer neuen Ordnung des Wissens: die Memex-Utopie (Bush 1945). Mit ihr suchte Vannevar Bush, die Verfügung über das zivilisatorische Wissen auf den Stand analog-elektronischer Technologie zu bringen. Ausgangspunkt der Überlegungen war nicht anders als bei Wells die Unangemessenheit der bestehenden Methoden der Wissensverwaltung aus der Frühzeit der Industrialisie-

22 Vgl. Freyermuth (2001*b*).

rung. Notwendig schien ihm Automatisierung im Umgang mit dem anschwellenden Wissen. Zu diesem Zwecke schlug Bush den *Memex* vor, eine Kombination von persönlicher Mikrofiche-Bibliothek und elektronischem Schreibtisch. Mit diesem *Memex* sollte jedes Individuum über das Anlegen eigener Verknüpfungsketten, so genannter *trails*, den epochalen Wissensbestand personalisieren können. Bush entwarf damit eine analoge Art Hyperlinks und mit ihnen eine neue, subjektivierbare flüssige Ordnung des Wissens. Zugleich mit dieser Personalisierung wollte er das zeitgenössische Wissen auch privatisieren, da ja jeder im eigenen *Memex* nach persönlichen Interessen modifizierbare Kopien speichern können sollte.

Die Memex-Utopie überschritt den technologischen Stand ihrer Zeit. Was sie mittels analoger Elektronik anstrebte, wirkte jedoch als Leitidee in der zweiten Jahrhunderthälfte fort. Wie Bush strebte auch Timothy Berners-Lee nach einer neuen Organisation des Wissens, um die mehr denn je anschwellende Informationsflut zu steuern. 1989 schlug er unter dem Titel „Information Management: A Proposal" ein digitales Publikationssystem vor, das eine dynamischere, benutzerfreundlichere Aufbereitung der Datenbestände leisten sollte (Berners-Lee 1989; Berners-Lee und Fischetti 1999). Das *World Wide Web*, das daraus entstand, leitete ein neues, auf digitalem Code basierendes Regime der Organisation und Distribution des zivilisatorischen Wissens ein, das den digitalen Möglichkeiten zur Virtualisierung von Dokumenten, ihrer entorteten Speicherung sowie arbiträren Multiplikation und damit dem Potenzial zur Privatisierung und Personalisierung verdateter Informationen entsprach – eine globale Ordnung des Wissens, die konsequenterweise nicht erst bei der Verwaltung analoger Artefakte einsetzte, sondern bereits beim Prozess der Publikation selbst, indem dieser entmaterialisiert, virtualisiert wurde.

Das *WWW* wurde so binnen weniger Jahre zur zentralen Region der digitalen Infrastruktur und damit unentbehrlich für alle Wissensarbeit. Insgesamt wächst zudem die Bedeutung dieser digitalen Infrastruktur kontinuierlich. Längst hat sich der Zugang zu ihr von einem Privileg oder Hobby zum Teil der kommunikativen Grundversorgung gewandelt. Je unverzichtbarer aber Software wird – und in je höherem Maße alles Wissen Software wird –, desto weniger kann der Mangel an demokratischer Kontrolle durch diejenigen bestehen, deren Existenz von ihr abhängt. Programme und Protokolle ermöglichen und limitieren zugleich zivilisatorische Erfahrungen. Was den Alltag so nachhaltig prägt, wovon auch das Überleben unzähliger Unternehmen und damit das Wohlergehen der Nationen abhängt, darf nicht auf Dauer Privatbesitz und außerhalb des Einflusses der Mehrheit der Anwender bleiben.

> „The brutal truth is this: when your key business processes are executed by opaque blocks of bits that you can't even see inside (let alone modify), you have lost control of your business. You need your supplier more than your supplier needs you—and you will pay, and pay, and pay again for that power imbalance. You'll pay in higher prices, you'll pay in lost opportunities, and you'll pay in lock-in that grows worse over time

> as the supplier (who has refined its game on a lot of previous victims) tightens its hold." (Raymond 1999, S. 152)

Hierarchische Bürokratien freilich kommen angesichts des akzelerierten Entwicklungstempos wie auch des gewandelten Demokratieverständnisses als Medium der notwendig werdenden öffentlichen Kontrolle kaum mehr in Frage. Diese Rolle fällt neuartigen beweglicheren, effektiveren und vor allem offeneren Strukturen zu. Für sie bietet die Open-Source-Praxis, wie sie Richard Stallman in den achtziger Jahren mit einer *General Public License (GPL)* zu kodifizieren suchte, ein Modell. Unter ihr wurden bis zur *Open Source Initiative* von 1998 so gut wie alle und werden auch heute noch viele freie Software-Programme veröffentlicht.

Grundgedanke der *GPL* ist das dem Copyright durchaus ironisch entgegengesetzte Copyleft. Es erlaubt, geistiges Eigentum beliebig zu modifizieren und auch profitabel zu vertreiben, solange das jeweils neue Produkt ebenfalls frei in dem Sinne bleibt, dass es Zugang zum Quellcode gewährt, damit andere es wiederum modifizieren können. Verboten sind also lediglich proprietäre Hinzufügungen, die für die Zukunft Peer-Review und Verbesserungen verhindern.

Mit diesem neuartigen Ansatz zur Regelung geistigen Eigentums schuf die *Free Software Foundation* ein folgenreiches Modell geistigen Gemeineigentums – einige Jahre, bevor das Internet zum Medium der Massen wurde und in ihm nicht nur Programme, sondern auch digitalisierte Texte, Töne und Bilder mehr oder weniger frei kursieren konnten. Seit der Wende zum 21. Jahrhundert, seit – spätestens mit *Napster* – die technische und damit praktische Unhaltbarkeit geltender Copyrightbestimmungen zu Tage trat und Fragen geistigen Eigentums zu einem zentralen Konfliktfeld der Digitalisierung avancierten, wurde die Hellsicht deutlich, mit der sich Stallman vor zwei Jahrzehnten bereits dem Problem stellte.[23]

Gleichzeitig aber erlaubt der historische Abstand auch, die Verhaftung der Free–Software-Bewegung im Zeitgeist und die daraus resultierenden Mängel ihrer Lösungsangebote zu erkennen. Nicht nur Stallmans damalige Äußerungen, auch die *GPL* war beeinflusst von Werten und Haltungen der alten Hippie-Hacker-Gegenkultur und ihrer zeittypischen antikapitalistischen Gegenkultur-Rhetorik. Charakteristisch für deren Ziele war der Schlachtruf des Hypertext-Erfinders Ted Nelson: „Computer Power to the People." Anfang der siebziger Jahre verlangte er nach einer radikalen Demokratisierung des bürokratisch-elitär geregelten Zugangs zu den wenigen und teuren Rechnern.

Diesen guten Zweck gedachte man allerdings in sozialistischer Tradition durch Vergesellschaftung des Vorhandenen zu erreichen. Zehn Jahre später, zur Zeit der Stiftungs-Initiative, äußerten sich die antikapitalistischen und eigentumsfeindlichen Wurzeln der Free-Software-Ideologie noch im changierenden Gebrauch des Adjektivs *frei*. Software sollte mal frei sein im libertären Sinne von freier Liebe und freier

23 Stallmans intellektuelles Wirken dokumentiert der von Joshua Gay (2002) herausgegebene Band „Free Software, Free Society".

Meinungsäußerung, mal frei im sozialistischen Sinne von Freikarten und Freibier – und in der Regel möglichst beides.

Zwar besaß die Forderung nach freierem Zugang zu Computern und Programmen in den sechziger und siebziger Jahren ihre Berechtigung, da digitale Technologien für Normalbürger praktisch unerschwinglich waren. Verstaatlichung oder staatliche Kontrolle, die in der industriellen Epoche übliche Lösung solcher Infrastrukturprobleme, bedeutete jedoch, den Teufel des wirtschaftlichen Monopols mit dem bürokratischen Beelzebub auszutreiben und technologischen Stillstand zu riskieren. Spätestens in der zweiten Hälfte der achtziger Jahre machte zudem die Durchsetzung des PCs als ganz normaler Konsumgegenstand die Forderung nach Gratis-Hard- oder Software praktisch gegenstandslos. Statt sich mit der Verwaltung vorhandenen Mangels aufzuhalten, hatte der Kapitalismus einmal mehr massenhafte Verfügung durch produktiven Zuwachs erwirtschaftet. Die Erfahrung steter Verbilligung von Hard- und Software ließ denn auch innerhalb der Free-Software-Bewegung in den neunziger Jahren die libertären Stränge über die sozialistischen siegen. Stallman etwa stellte klar, dass die Distributionsregeln, für die sich die Stiftung einsetzte, lediglich frei im Sinne von freier Rede meinten, nicht aber unbedingt frei im Sinne von Freibier (Free Software Foundation 2006).

So verstanden bietet die Open-Source-Praxis, wie Johnson (1998) in *Feed* darlegte, einen besseren Ansatz für eine zeitgemäße Form demokratischer Kontrolle der essentiellen digitalen Infrastruktur und der Produkte, die ihr die Existenz verdanken. Denn die meisten Programme – wie im Übrigen viele andere Varianten intellektueller Schöpfung beziehungsweise geistigen Eigentums, von Filmen bis zu Forschungsprojekten – entstehen heute als Ergebnis einer Vielzahl von Einzelanstrengungen. Besitztitel einzelner werden ihrer Genesis nicht gerecht. Zudem verhindern sie die freie Weiterentwicklung durch interessierte Gruppen, selbst durch die ursprünglichen „Mitarbeiter". Gegenüber vielen avancierten Produkten der Wissensgesellschaft versagt so die Copyright- und Patent-Haltung zum Eigentum. Open-Source-Regelungen hingegen sorgen „for a more democratic version of software development, where good ideas are shared by the collective rather than being hoarded as proprietary intellectual capital—democratic, in other words, in both the creation and the use of information." (Johnson 1998).

Das Copyleft der *GPL* bietet so ein sowohl ethisch wie wirtschaftliches interessantes Eigentumsmodell für Produkte vernetzter Wissensarbeit. Was nämlich unter einer Perspektive, die vom einzelnen Schöpfer ausgeht, wie eine Enteignung aussieht, bewirkt im Falle kollektiver Kreation gerade das Gegenteil: indem es die ebenso übliche wie unangemessene individuelle Aneignung gemeinschaftlich geschaffener Werte durch private oder juristische Personen verhindert. Gegen den Einwand von *Microsofts* Vizepräsident Craig Mundie, die *GPL* entwerte die Idee kreativer Arbeit, wendete denn auch Tiemann (2001) ein, dass dies genauso wenig der Fall sei, wie die Abschaffung der Sklaverei einst die Idee des Privateigentums schmälerte: „As software becomes more

and more a part of our life, the foundation of our communication, and a representation of our personal identities, shouldn't we, the people, be the ones who have the freedom and control over the software (and not the other way around)?" In der Wissensgesellschaft gehört, wie Lawrence Lessig im Microsoft-Monopol-Prozess deutlich machte, eine demokratische Kontrolle der digitalen Infrastruktur unabdingbar zur Erhaltung intellektueller und wirtschaftlicher Freiheit.[24]

Viele mögen noch nicht das Maß erkennen, in dem ihr geschäftliches und auch privates Wohlergehen von digitalen Programmen abhängt, die heute noch in der Regel proprietärer Willkür unterliegen. Insbesondere die Verschmelzung von Informatik und Genetik deutet an, wie existenziell Open-Source-Regelungen für die Wohlfahrt der gesamten Menschheit werden könnten. Bei einer Vielzahl teils rechtlich geschützter, teils geheim gehaltener Erkenntnisse geht es bereits um genetisches Wissen, um nichts anderes als den – dergestalt proprietär werdenden – Quellcode unseres Lebens. Auf dem Gebiet der Bioinformatik braucht es daher wie auf keinem anderen freie Verfügbarkeit von Informationen.

Permanenter Peer-Review kann Missbrauch verhindern und – wichtiger noch – die Forschung beschleunigen. Proprietärer Besitz hingegen, der die Mehrheit der Wissenschaftler von der Mitarbeit ausschließt, kostet Zeit und damit Leben. „We can't afford the equivalent of a biological Microsoft", sagen Experten wie Drew Endy vom *Molecular Sciences Institute* in Berkeley (Philipkoski 2001).

Die Alternative, eine bessere Möglichkeit zur Nutzung unseres biologischen Gemeineigentums beziehungsweise des Wissens, das wir im Begriff sind, davon – über uns – zu erwerben, deutet dagegen seit den frühen neunziger Jahren die Erfolgsgeschichte von Linux an. Sie allerdings verdankte sich der nächsten Stufe digitaler Vernetzung.

8 Dritte Phase: Soziale Durchsetzung der Open-Source-Praxis

Um 1990 wurde mit neuen Prozessoren von *Intel* und *Motorola* und vergleichbaren Fortschritten in den Speichermedien erstmals digitales Gerät für Privatleute erschwinglich, das so leistungsfähig war wie zuvor nur Mikrocomputer zum Preis eines Mittelklassewagens. Gleichzeitig setzte das Internet zu seinem bis dato größten Entwicklungssprung an. Ermöglicht wurde er wiederum von freier Software.

Timothy Berners-Lee, damals am CERN-Forschungsinstitut in Genf, heute *MIT*, schuf eine Hypertextmaske, die das *World Wide Web* begründete, und veröffentlichte die Software 1991 in akademischer Tradition – ohne jeden Copyright-Anspruch, als ein Geschenk an die wissenschaftliche Gemeinde und mit der direkten Aufforde-

24 Darstellungen des Prozesses bzw. der Entwicklung, die Lessig zu seinen Einsichten brachte, geben etwa Auletta (2001) und Levy (2002). Lessig (1999) selbst hat die Bedeutung einer demokratischen Kontrolle der digitalen Infrastruktur in seinem grundlegenden Buch „Code and Other Laws of Cyberspace" dargelegt.

rung, seine Erfindung nach Möglichkeit zu verbessern. Der Erfolg verdankte sich wesentlich diesen Umständen. Binnen weniger Jahre verdrängte das *WWW* andere zeitgenössische Ansätze, das Netz benutzerfreundlicher zu machen, etwa das um 1990 erfolgreiche Gopher-System, auf das jedoch 1993 die Universität von Minnesota kommerzielle Ansprüche erhob:

> „This was an act of treason to the academic community and the Internet community. Even if the university never charged anyone a dime, the fact that the school had announced it was reserving the right to charge people for the use of the gopher protocols meant it had crossed the line." (Berners-Lee und Fischetti 1999, S. 73)

Das *World Wide Web* profitierte von diesem Missgriff und wurde unter den Wissenschaftlern, Technikern, Hobbyisten und Künstlern, die damals zu den Hauptnutzern des Internets zählten, schnell zu dessen beliebtester Region.

Diese erste Zeit aber, in der das *WWW* neben dem wissenschaftlichen Austausch vor allem subkultureller Kommunikation, den Experimenten der Avantgarden und persönlicher Selbstdarstellung gehörte, währte kaum zwei, drei Jahre. Die Privatisierung des Zugangs über kommerzielle *ISP*s, die Einführung niedriger Pauschalgebühren sowie die sich kontinuierlich vereinfachende Personalisierung des Gebrauchs durch schnellere und stabilere Browser ließen in den frühen neunziger Jahren Tag für Tag Zehntausende neuer *netizen* zuwandern. Die Besiedlung des Datenraums beschleunigte sich exponentiell. Der gemessene Datenverkehr etwa verdoppelte sich zwischen 1989 und 1995 alljährlich; danach, trotz der ja weit erhöhten Ausgangsbasis, gar binnen sechs bis neun Monaten (Colwell 1999).

Das *WWW* verwandelte so binnen weniger Jahre das Internet von einem Medium der Eliten zu einem Medium der Massen. Für diesen Erfolg gab es vor allem drei Gründe: Ein erster Vorteil gegenüber den älteren Zonen des Internets bestand in dem transmedialen Charakter des *WWW*. Die grafische Gestalt, die es dem Datenraum verlieh und die ihrem Wesen nach hyperrealistisch war, besorgte eine räumliche Konkretisierung. Deren Faszination auf die Zeitgenossen drückte sich in der Beliebtheit von Formulierungen wie *Cyberspace* und *Surfen* aus. Erst das *WWW* machte den Daten-Nichtort sinnlich besetz- und besiedelbar und eröffnete damit in ihm zweitens eine Vielzahl virtueller Gegenöffentlichkeiten. Hatte bereits das Internet den Peer-Review auf Echtzeit beschleunigt und damit die technische Voraussetzung für entortete interaktive Kollaboration und Kreation geschaffen, so besorgte das *WWW* nun die Demokratisierung des Verfahrens.

In der ersten Hälfte der neunziger Jahre wandelte sich der PC damit vom Arbeitsgerät Einzelner – primär genutzt für Schreiben, Rechnen, Verwalten – zum vernetzten Kommunikations- und multimedialen Unterhaltungsgerät. Mit dem neuen, leistungsfähigeren Technologie-Ensemble aus Multimedia-PC und digitaler Vernetzung wuchs auch der Bedarf nach entsprechend professioneller und dennoch erschwinglicher Soft-

ware. Stallmans GNU-Unternehmen wie auch andere experimentelle Anläufe, *UNIX* zu portieren, waren jedoch noch nicht sehr weit gediehen. Vor diesem Hintergrund kam es schließlich nach 1991 zum nächsten Entwicklungssprung virtueller Kollaboration.

Sein Initiator war Linus Torvalds. Schon für die Anfänge seines offenen Betriebssystems spielte das Internet eine wesentliche Rolle: Als Student an der Universität von Helsinki nutzte Torvalds auf seinem Heimcomputer *386-Minix*, eine experimentelle Version des auf *UNIX* basierenden Lehrbetriebssystems *Minix*. Um von seinem PC aus Usenet-Nachrichten oder E-Mail auf dem Universitätsserver lesen zu können, schrieb er eine eigene Terminalemulation. Um wiederum die zu nutzen, bedurfte es eines Neustarts, nach dem die üblichen Minix-Features nicht mehr zugänglich waren. Aus Torvalds Anstrengungen, das eigene Programm mit den Features zu verbinden, die *Minix* bot, entstand – unter Benutzung vorhandener Elemente der GNU-Variante – am Ende der Linux-Kernel.[25] Glynn Moody (2001) betont in seinem Buch die historische Ironie: Weil Torvalds nicht ins Internet konnte, entwarf er ein Betriebssystem, das dann nur dank Internet überhaupt reifen konnte.

Mit der Eröffnung des *WWW* – das auf Software beruhte, die ihre Gestalt in virtueller Kollaboration fand – endete so die zweite Phase der Open-Source-Praxis, in der deren Anstrengungen sich noch primär auf die Vernetzung selbst gerichtet hatten. Die frühen neunziger Jahre sahen den Beginn der dritten Phase. Sie kennzeichnete eine doppelte Akzentverschiebung. Zum einen weitete sich mit der Globalisierung der Vernetzung die virtuelle Kollaboration über die Grenzen der USA aus; sowohl Berners-Lee und das *WWW* wie auch Torvalds und Linux gaben dafür ein Beispiel. Zum anderen verlagerte sich der Schwerpunkt der inhaltlichen Interessen – des Ungenügens der Nutzer also – von der Vernetzung selbst auf Betriebssysteme und Programme.

Deren erfolgreiche Produktion unter den neuen Bedingungen globaler Kollaboration verlangte jedoch über die Organisation der Ver-Öffentlichung und Nutzung von Arbeitsergebnissen hinaus, wie sie Stallmans *GPL* seit Ende der achtziger Jahre bot, auch nach einer effektiveren Organisation der kreativen Arbeit selbst.

9 Die fünfte Innovation: Vernetztes Wissensarbeiten

Die Programmierer für Linus Torvalds' Projekt eines UNIX-Kernels für 386er-Prozessoren rekrutierten sich nicht länger regional, sondern international aus der explosiv wachsenden Online-Bevölkerung. Was sie in so überraschend großer Zahl zu Linux zog, beschreibt Eric Raymond als *social engineering*: „The most important feature of Linux, however, was not technical but sociological" (Raymond 1999). In der freien

25 Linux steht für *Linus' UNIX*. Gegner des Personenkults verstehen es auch als *Little UNIX* im Unterschied zum großen Mikrocomputer-Bruder. Die GNU-Bewegung inspirierte Torvalds dazu, seine Software unter die *GPL* zu stellen. Darüber hinaus schuf das GNU-Projekt die Systemtools und Anwendungen, die Linux erst zu einem benutzbaren Betriebssystem machten – weshalb viele bis heute statt der einfachen Bezeichnung Linux den genaueren Terminus GNU/Linux bevorzugen.

Softwarebewegung hatte man zwar schon vorher das Internet zur Kommunikation und Übermittlung von Code genutzt. Doch die Organisation größerer Projekte hatte sich den neuen technischen Möglichkeiten kaum angepasst. Bei aller geografischen Zerstreutheit operierten die Beteiligten nach dem Vorbild industrieller Arbeit in recht geschlossenen Hierarchien. Arbeitsergebnisse wurden nur innerhalb der Gruppen ausgetauscht und Programme erst nach Fertigstellung veröffentlicht. Torvalds brach mit diesen Gepflogenheiten, und der überraschende Doppelerfolg von *WWW* und Linux, von Software, die nach 1990 in virtueller Kollaboration erzeugt wurde und dann im Laufe nur weniger Jahre die Arbeit und den Alltag von Hunderten von Millionen Menschen nachhaltig beeinflusste, deutet einmal mehr auf tiefergehende zivilisatorische Veränderungen. Sie stellten zunächst die etablierten Verfahren der Wissensproduktion und Wissensverwaltung in Frage, betrafen dann aber auch generell die tayloristische Arbeitsorganisation und damit den Kern des Industrialismus. Auch hier kann der historische Vergleich den Wandel erhellen, der am Ende des 20. Jahrhunderts einsetzte.

Das von Frederick Winslow Taylor um 1900 begründete Verfahren der wissenschaftlichen Betriebsführung hatte den Menschen nicht länger als eigenständig Handelnden begriffen, sondern als Teil der industriellen Maschinerie. Bewegungsabläufe der Arbeitenden wurden vermessen und durch Zerlegung und Umorganisation optimiert. Indem der Taylorismus den menschlichen Körper zum Anhängsel der Maschine machte, steigerte er die mechanische Effizienz – und vernichtete psychische Motivation und Kreativität. Das Buch, in dem Taylor (1911) seine „Principles of Scientific Management" zusammenfasste, wurde ein internationaler Bestseller und Taylor – zusammen mit Henry Ford, der den Taylorismus wiederum optimierte – zum Geburtshelfer der Arbeitswelt des 20. Jahrhunderts. Denn den Arbeitenden ihre Zeitsouveränität zu rauben, war ein epochaler Akt.[26]

Agrarische und handwerkliche Produktion orientieren sich nicht am Abarbeiten von Zeit, sondern an Aufgabenerfüllung wie Aussaat und Ernte oder der Erledigung von Aufträgen. Was Jahrtausende die Regel gewesen war – dass jeder sich die eigene Arbeit relativ selbstständig nach sachlichen Erfordernissen einteilte –, wurde dank Taylor zum Privileg und blieb die Regel allein in Nischen des hoch qualifizierten Handwerks und künstlerischer oder wissenschaftlicher Tätigkeiten. Unter industriellen Bedingungen wandelte sich Arbeitszeit für die Mehrheit der Menschen zur Fremdzeit, zu einem Teil des Tages, an dem physische Anwesenheitspflicht herrschte und erwachsene Bürger durch Anweisungen und Beaufsichtigung systematisch entmündigt wurden. Taylor selbst sprach von einer „mental revolution", einer Umwälzung der herrschenden Mentalität.

Sie war außerordentlich folgenreich. Binnen weniger Jahrzehnte stieg der Taylorismus zu einer dominierenden Zivilisationstechnik auf, die bald auch die Arbeit in

26 Vgl. zum Folgenden Kanigel (1997) sowie Landes (1969) und Mokyr (1990). Speziell zur Mechanisierung, Industrialisierung und dann Digitalisierung der Zeit vgl. Landes (1983); Kern (1983); Gleick (1999) sowie als Einführung Freyermuth (2000).

nichtindustriellen Bereichen veränderte, vom Militär bis zur zivilen Verwaltung mit ihrem Informationsfluss nach dem Fließbandmodell, von der Arbeit in den Traumfabriken bis zur Forschung in den Denkfabriken, von der Erledigung der Hausarbeit bis zur Organisation des eigenen Hobbys und privaten Konsums. Die Abläufe in einem Supermarkt des 20. Jahrhunderts etwa verhalten sich zu denen, die im Einzelhandel des 19. Jahrhunderts üblich waren, wie die taylorisierte Fabrik zur handwerklichen Produktion. Ebenso prägte die Erfahrung der Taylorisierung Kultur und Unterhaltung der Moderne. Spuren finden sich etwa, wie die Literaturkritik erkannte, in der „taylorisierten" Sprache Hemingways oder Chandlers. Und sie finden sich vor allem im populärsten industriellen Medium, dem Film – in seiner Ästhetik der durch Einzelaufnahmen zerhackten, zerschnittenen und dann ummontierten Bewegungsabläufe und auch in seiner zu industriellen Zeiten vorherrschenden Rezeptionsform, dem Kino, in dem die schnell laufenden Filmbilder dem Publikum genauso die Zeitsouveränität rauben wie das Fließband der Fabrikarbeiterschaft.

Der praktische Erfolg seiner empirisch-systematischen Methoden sicherte Taylor die unverbrüchliche Feindschaft ganzer Generationen von Soziologen und Kulturkritikern. Der Taylorismus und seine konsequente Fortführung, der Fordismus samt Fließband, wurden zum Inbegriff dessen, was am industriellen System falsch war: der Unterwerfung der Individuen unter den Rhythmus der Apparate und hierarchisch–zentralistische Kontroll- und Kommandostrukturen, die auf inhaltliche wie zeitliche Fremdbestimmung von jedermanns Handeln zielten.

Die industrielle Ordnung der Arbeit machte sie zur Last – weshalb ihre Verwalter auf eine Vielzahl von Kontroll- und Zwangsmaßnahmen sowie auf eine stete Steigerung der finanziellen Anreize angewiesen waren. Unter analogen Bedingungen ließ sich daran kaum rütteln. Alle utopischen Gegenbewegungen scheiterten an den Sachzwängen industrieller Technologie.

Erst mit der Digitalisierung schuf sich die Menschheit die nötigen Voraussetzungen, um das Joch industrieller Arbeit abzuschütteln. Das Linux-Projekt war in den neunziger Jahren dafür wegweisend, indem es die globale Vernetzung nicht länger nur zur Kommunikation, sondern für eine prinzipielle Umorganisation der Arbeit nutzte. Torvalds verzichtete so weitgehend wie niemand vor ihm auf die Zwangsjacken der industriellen Tradition. Es gab kaum lineare Abschnitts- und Terminplanungen, keine hierarchischen Vorgaben und restriktiven Zugangskontrollen. Grundsätzlich konnte sich jeder beteiligen. Neuen Code testete Torvalds nicht lange intern, er veröffentlichte ihn vielmehr in schneller Folge im Internet. Das Auffinden von Fehlern und ihre Beseitigung blieb damit weitgehend der globalen Gemeinschaft überlassen. Mit dem Fortschreiten des Projekts kam es zudem zu einer modularen Strukturierung, die Verantwortung für Teilbereiche auf interessierte Subgruppen übertrug. An jeder Aufgabe arbeiteten jeweils Menschen, denen ihr Part nicht zugeteilt worden war, sondern die ihn sich ihren Qualifikationen und Motivationen entsprechend selbst ausgesucht hatten.

Ausgehend von der Tradition der Anwender-Selbstermächtigung und aufbauend auf den Betriebssystemen und der digitalen Infrastruktur, die freie Software in den siebziger und achtziger Jahren geschaffen hatte, setzte Torvalds so abstrakte digitale Werte wie Dezentralisierung und Enthierarchisierung in praktisches Handeln um, insbesondere bei der Rekrutierung von Arbeitskraft auf dem Wege globaler Selbstselektion und der Gewährung zeitlicher Selbstbestimmung. Nicht die Arbeitsteilung, wohl aber die Arbeitszuteilung wurde aufgehoben. An die Stelle von überwachender Kontrolle trat Eigenverantwortlichkeit in autonomen Teilbereichen.

Dieser sozialen Innovation verdankte sich der enorme Anklang, den Torvalds Bitte um Mithilfe fand. Globale Offenheit statt lokaler Geheimniskrämerei, Einbeziehung Interessierter statt Ausgrenzung fremder Hilfe, dynamische Selbstorganisation statt statisch-linearer Planvorgaben, Zeitsouveränität statt Fremdbestimmung, Eigenverantwortlichkeit statt verwalteter Kontrolle, permanente Peer-Reviews statt Beurteilungen durch Vorgesetzte und Management – mit dieser gebündelten Abkehr vom Vorbild industrieller Arbeitsorganisation entwarf das Linux-Projekt bereits in seiner Frühzeit ein Modell vernetzter Wissensproduktion. Richard Gabriel von *SUN* beschreibt es als: „a kind of semi-chaotic, self-organising behaviour in which numerous small acts of repair can lead to quickly built, complex and massive creations" (The Economist 2001).

Neben der optimalen Nutzung der technischen Produktivkräfte zur entorteten Echtzeit-Kollaboration war die verbesserte Nutzung der knappen Ressource menschlicher Kreativkraft – die gesteigerte Motivation der Mitarbeiter – für den Erfolg entscheidend. Denn die Linux-Praxis gewährte Arbeits- und Gemeinschaftserfahrungen, die im Gegensatz nicht nur zur verwalteten kommerziellen Softwareproduktion, sondern auch zur bis dahin üblichen Organisation freier Projekte standen. Die Befriedigung, die viele in ihrer Arbeit fanden, erinnerte vielmehr an die durch wissenschaftliche oder künstlerische Arbeiten. Sie werden ja gleichfalls nicht fremdbestimmt, zu festen Arbeitszeiten und allein zum Zwecke des Lebensunterhalts ausgeübt. Bei ihnen geht es wesentlich auch um das eigene Selbstwertgefühl und Ansehen in der Gemeinschaft.

Torvalds schuf so nichts weniger als eine neue Organisationsform der Arbeit, die als erste um das Internet zentriert war und dessen einzigartige Möglichkeiten zu einer entorteten Echtzeit-Kollaboration nutzte. Dieser Reorganisation der Arbeitsabläufe, mit der die Linux-Praxis an den Wurzeln des industriellen Systems rüttelte, korrelierten gleichzeitig Anstrengungen, mit der Kollaboration der Individuen auch deren Konkurrenz effektiver zu gestalten.

10 Die sechste Innovation: Konkurrierende Kollaboration

Auf den Begriff gebracht werden sollte, was in den frühen neunziger Jahren gewissermaßen naturwüchsig aus dem akademischen Hackermilieu an neuen, von digitaler

Technik informierten Verhaltens- und Denkweisen erwuchs, freilich erst am Ende des Jahrzehnts, vor allem natürlich in Eric S. Raymonds Essays, die als Manifest der Open-Source-Bewegung wirkten (Raymond 1999). An die Stelle der kapitalistischen Kontrollwirtschaft trete, schreibt Raymond, ein digital ermächtigtes Modell der Arbeit, das lineare Planungshierarchien, wie sie beispielsweise den Bau einer Kathedrale prägen, durch die offenen, von Gleichberechtigung ausgehenden Transaktionsstrukturen eines Basars ersetzt.

Raymonds Thesen, die er auf Kongressen vortrug und im Internet publizierte, fanden breiten Anklang – nicht zuletzt, weil sie der zunehmend erfolgreichen Praxis freier Software ein theoretisch-historisches Fundament verliehen und dabei gleichzeitig den ideologischen Ballast abwarfen, der Stallmans Free-Software-Bewegung gehindert hatte, breitere Akzeptanz zu finden.

Historische Vorbilder solch kollaborativer Produktion, die schon zu analogen Zeiten auf die Open-Source-Praxis vorauswiesen, fanden sich in den Experimenten der künstlerischen Avantgarden des 20. Jahrhunderts, etwa in Techniken der Appropriation – der kreativen Aneignung fremden Materials beim Collagieren und Sampling – und vor allem in den Schreibspielen der Surrealisten. Diese Nähe der Open-Source-Praxis zur Kunst lässt sich nicht nur im historischen Rückblick feststellen. Sie prägte durchaus die zeitgenössische Praxis. Davon zeugen die Selbstäußerungen führender Vertreter der Free-Software- und Open-Source-Gruppen, die immer wieder von Selbstausdruck und Selbstverwirklichung sprechen. Eric Raymond zum Beispiel sagt: „I do what I do primarily for artistic satisfaction, and what I want is to know that other people consider it good art.“[27]

Zu den prominentesten Lesern, die sich von Raymonds Schriften beeinflussen ließen, gehörte Netscape-Gründer Jim Barksdale. Er fasste den Entschluss, mit der industriellen Geheimniskrämerei zu brechen und den Quellcode des Netscape-Browsers zu veröffentlichen. Ironischerweise popularisierte die Firma, deren Börsengang keine drei Jahre zuvor das Internet als Geschäftsgelegenheit ins Bewusstsein der breiten Öffentlichkeit katapultiert hatte, damit auch Anfang 1998 die Gegenbewegung. Nach Jahren eher verborgenen Wirkens gerieten die Vertreter der freien Software ins Licht der Medien. Raymond, Bruce Perens und andere Protagonisten der Open-Source–Bewegung beschlossen, die Gunst der Stunde zu nutzen. Bei einem eilig einberufenen Kongress in Palo Alto suchten sie nach einem pragmatischen Slogan, einem treffenden Markenzeichen für die Bewegung, das anders als der von Stallman, dem Kommunarden des Code, geprägte Begriff der *Free Software* nicht mehr die sozialistischen Träume der analogen Gegenkultur der sechziger Jahre assoziieren ließ.

Mit *Open Source* fanden sie die magische Wortkombination. Der Begriff war mehr als eine unverdächtige und mehrheitsfähige Sprachregelung. Er erfasste, was sich durch Linux verändert hatte. War *Free Software* der Kampfbegriff einer kleinen Minderheit gegen die erdrückende und Veränderungen unterdrückende Übermacht

27 Zitiert nach Leonard (1998).

proprietärer Produkte gewesen, so reagierte die konsensträchtige Rhetorik der Open–Source-Bewegung auf die Realität des Jahres 1998: auf den wachsenden Erfolg bei avancierten Anwendern und den allmählichen Aufstieg der freien Software vom subkulturellen Billigparia zum technologisch ernst zu nehmenden Konkurrenten und vor allem auch zum leistungsfähigen Wirtschaftspartner. Für den weiteren Erfolg erwies sich dann die 1998 geschaffene intellektuelle und institutionelle Open-Source-Plattform als Geburtshelfer. Sie ebnete der freien Software den Weg in die Geschäftswelt. Bald gehörten zu den Verbündeten mächtige Konzerne wie *Intel*, *Oracle*, *Dell*, *Novell* und vor allem *IBM*. Der Einsatz der etablierten Firmen für Open-Source-Software – neben Linux insbesondere auch für *Apache* – sowie die gelegentliche Freigabe von eigenem Quellcode (*outsourcing to open source*) erhöhte die Akzeptanz von freier Software und verbreiterte die Nutzerbasis.

Die neuen Weggefährten der Open-Source-Bewegung orientierten sich dabei deutlich an dem Vorbild, das jene kleinen Firmen gegeben hatten, die bereits zwischen 1992 und 1994, dem Jahr, als Linux 1.0 veröffentlicht wurde, rundum die freie Software entstanden waren. Deren Geschäftsmodell differierte zwangsläufig von dem traditioneller Softwarefirmen. Im Verhältnis untereinander – des einen zu den anderen Open-Source-Anbietern – wich die übliche Konkurrenz mittels proprietärer Angebote einer so genannten *coopetition*, einer Mischung aus Zusammenarbeit (*cooperation*) in technologischer Hinsicht und Konkurrenz (*competition*) im serviceorientierten Wettbewerb um die Kunden. Diese Konzentration auf Service wiederum beruhte auf der Einsicht, dass Software – die immaterielle Abspaltung der in mechanischen und industriellen Maschinen materiell gefangenen Algorithmen – ihrem Charakter nach eben kein abgeschlossenes industrielles Produkt darstellt, sondern vielmehr einen prinzipiell nach hinten offenen Prozess. Jede Version ist nur eine Stufe in einem steten Entwicklungsvorgang, der auf die Befriedigung der sich wandelnden Ansprüche der Anwender zielt. Auch gratis erhältliche Open-Source-Software bietet daher reichlich Service- und damit Geschäftsgelegenheiten. Firmen wie *SUSE*, gegründet 1992, oder *Red Hat* und *Caldera*, beide 1994, sammelten aus der Menge freier Software, die sich im Netz akkumulierte, funktionierende Pakete zusammen, sorgten für die Komplettierung des einen oder anderen fehlenden Elements und verliehen ihrem Auswahlangebot durch Supportleistungen gesteigerten Gebrauchswert.

Gewissen ökonomischen Rückschlägen im Gefolge der Dotcom-Krise zum Trotz, hat sich die Neunziger-Jahre-Erfolgsgeschichte von Linux und anderer offener Software im 21. Jahrhundert fortgesetzt. Eric Raymond behauptet denn auch einen doppelten historischen Trend zu offenen Systemen (Raymond 1999, S. 146 ff.). Er zeige sich zum einen im Zyklus einzelner Programme, die mit zunehmendem Alter gegenüber innovativer Konkurrenz nur bestehen können, indem sie zu De-facto-Standards avancieren. Als Beispiel nennt er *Doom*: Die Veröffentlichung des Quellcodes zog eine Vielzahl freier Entwickler an, die dem Spiel neues Leben einhauchten und so seine Auswertungszeit verlängerten. Zum anderen aber tendiere die Digitalisie-

rung grundsätzlich zur Etablierung offener Standards. Vom Fünfziger-Jahre-Chaos proprietärer Betriebssysteme und Programme, deren Daten nicht einmal zwischen Computermodellen desselben Herstellers austauschbar waren, führe so über proprietäre Standardisierungen, wie sie *Microsoft* leistete, eine gerade Entwicklungslinie zu offenen zivilisatorischen Standards, wie sie sich mit Linux ankündigen.

11 Resümee und Ausblick: Digitale Avantgarde

Was wir heute als Open-Source-Praxis kennen, entfaltete sich im Laufe der zweiten Hälfte des 20. Jahrhunderts in drei Stufen. Die erste lässt sich als Vorgeschichte begreifen: In den späten fünfziger und frühen sechziger Jahren kam im akademischen Milieu bei dem Umgang mit digitaler Technologie der Wunsch auf, die aus der industriellen Praxis resultierende Fremdbestimmung zugunsten individualisierender Ermächtigung zu überwinden. Die avantgardistischen und zunächst angesichts des Stands digitaler Technik eher utopischen Ansprüche sukzessive zu realisieren und gleichzeitig zu popularisieren, gelang dann im Kontext des Ausbaus digitaler Kommunikationstechnik durch die Überwindung regionaler Isolierung und Etablierung erst nationaler, dann globaler Echtzeit-Kollaboration. Ausgehend von dem Wendepunkt der Eröffnung des *Arpanet* wurden die siebziger und achtziger Jahre zur Gründerzeit nicht nur digitaler Vernetzung, sondern auch der Open-Source-Praxis. Deren volle Entfaltung sahen dann – ausgehend vom nächsten Wendepunkt, der Eröffnung des *WWW* – die neunziger Jahre. An den Ursprung der Open-Source-Praxis in isolierten subkulturellen Gruppen schloss sich so als erste Phase ihre nationale Etablierung durch wissenschaftliche Eliten zur Zeit institutioneller digitaler Vernetzung an und dann die zweite Phase ihrer globalen Popularisierung zur Zeit der Privatisierung digitaler Vernetzung.

Während dieser drei Stufen – Ursprung, erste Phase der Begründung, zweite Phase der Popularisierung – folgte die Entwicklung einer Reihe sich sukzessiv entwickelnder und bis heute fortwirkender Interessen und Prinzipien:

Selbstermächtigung Am Anfang stand der Wille zu einer Souveränität im Umgang mit Mitteln und Artefakten, wie sie zuvor Einzelnen – Bedienern, Rezipienten, Konsumenten – unmöglich gewesen war. Denn was zu industriellen Zeiten fortschrittlich war und auf nahezu alle Bereiche der Produktion, selbst auf Verwaltung, Schulen und Universitäten abfärbte – hierarchisch-zentralistische Kontroll- und Kommandostrukturen, die auf inhaltliche wie zeitliche Fremdbestimmung von jedermanns Handeln zielen –, bremste nun die gesteigerten produktiven und kreativen Möglichkeiten digitaler Technologie. Eine ähnliche Pionierrolle, wie sie einst die Autoindustrie in der Überwindung handwerklicher und agrarischer Organisationsformen der Arbeit erfüllte, fiel daher seit der Mitte des 20. Jahrhunderts den aufsteigenden digitalen Hard- und Software-Branchen zu. Die Geltung ihrer Organisations- und Managementmodelle, schreiben etwa

Don Tapscott und David Ticoll (2000), „isn't confined to the ones-and-zeros world of software. With the Internet's arrival, astute companies can capitalize on many aspects of the open-source philosophy to gain competitive advantage in the development of products as solid as a new car".[28]

Nutzergetriebene Evolution von Standards Unter industriellen Verhältnissen beschränkte sich der Prozess der Selektion am Markt auf die Selektion kompletter Produkte. Mit der Digitalisierung wurde Einflussnahme auf die Produktion und Modifikation der Produkte selbst möglich: Wo zuvor die Wahl zwischen fertigen Waren war, wurde durch die unmittelbare Einbringung von Verbraucherinteressen und Anwenderwissen eine nutzergetriebene Entwicklungsarbeit bis hin zur Selektion einzelner Features möglich. Open-Source-Strukturen dehnten so seit den siebziger Jahren die Wirkung der Marktmechanismen auf den Entwicklungsprozess aus. Dadurch eskalierte – da Kompatibilität ein Basisinteresse aller Nutzer ist – der Standardisierungsdruck.

Produktionsgemeinschaft der Gleichen Zur Zeit analoger Medien dienten Peer-Reviews den wissenschaftlichen Eliten als Medium produktiver Kritik. Die Etablierung multimedialer Echtzeitinteraktion durch digitale Vernetzung verwandelte diese Kommunikation der Gleichen in ein Mittel kreativer Wissensproduktion und demokratisierte seinen Gebrauch zugleich. Die Praxis des Peer-Review drang aus der Forschung in die breitere Öffentlichkeit. Hersteller wie Händler etwa, die seit den neunziger Jahren gezielt Foren zum Peer-to-Peer-Austausch der Nutzer und zur interaktiven Wissensproduktion einrichteten oder existierende Foren beobachteten und an ihnen partizipierten, gewannen mit wachsender Vernetzung ihrer Kunden einen klaren Wettbewerbsvorteil gegenüber anderen, die auf dem hierarchisch-analogen Modell von Kundendienst und technischer Hilfeleistung beharrten.

Geistiges Gemeineigentum Das Verständnis geistigen Eigentums wurde – seit die Vorstellung von ihm gewissermaßen mit dem Buchdruck aufkam – durch seine materielle Repräsentation geprägt, also die mechanische Reproduktion und analoge Distribution über Raum und Zeit. Die gegenwärtige Ersetzung analoger Technologien durch digitale stellt diesen seit der frühen Neuzeit etablierten Umgang mit geistigem Besitz in Frage. Nicht nur fallen zunehmend die materiellen Kontrollmechanismen aus. Obendrein bremst das überholte System der Eigentumssicherung heute schon den digitalen Fortschritt so nachhaltig, wie vor 200 Jahren umgekehrt der Mangel an Eigentumsgarantien den industriellen Fortschritt hemmte. Free-Software- und Open-Source-Bewegung zielen insofern – wie in der industriellen Frühzeit die Kombination von Urheberrechtschutz und Ver-Öffentlichtung zuvor privater Bibliotheksbestände – auf

28 Vgl. auch Tapscott et al. (2000).

einen nun der digitalen Epoche angemessenen Kompromiss zwischen der Sicherung geistigen Eigentums und seiner möglichst ungehinderten Nutzung. Die praktische Relevanz solcher Vermittlung geht weit über den in der Diskussion dominierenden Komplex von Raub- und Privatkopien intellektueller und künstlerischer Werke hinaus. Im wirtschaftlichen Alltag etwa stellt sich die Frage nach dem Eigentum an und dem Umgang mit den Daten, die bei der Virtualisierung von Transaktionen automatisch anfallen. Längst gehören denn auch interaktive Echtzeitzugriffe auf Informationen, die vor wenigen Jahren noch als „Firmendaten" geschütztes Herrschaftswissen geblieben wären – bekannte Mängel und Inkompatibilitäten eines Produktes, Einschätzungen und Beschwerden anderer Nutzer, Lagerbestand, Abwicklungsstand, Verfolgung des Transportweges usw. – zum online üblichen Auskunfts- und Serviceangebot.

Vernetztes Wissensarbeiten Der Einsatz digitaler Techniken zur Etablierung von Kollaborationsstrukturen ermöglichte seit den neunziger Jahren in der Open-Source–Praxis zeitliche Autonomie und Prozesse der Selbstselektion. Zu den Resultaten gehörten bessere Motivation und gesteigerte Kreativität. Damit deutet sich die Auflösung hierarchischer Strukturen und eine grundsätzliche Abkehr von dem an, was der IT-Experte und Gewerkschaftler Ulrich Klotz (1999) einmal die „Misstrauenskultur des Industriesystems" nannte. Der innovative Teil der Open-Source-Praxis, dessen Pionier Linus Torvalds war, scheint zunächst zwar primär ein Modell für die bessere und menschlichere Organisation kreativer Wissensarbeit. Inwieweit es sich auf andere Bereiche übertragen lässt, muss die Zukunft noch zeigen. Doch an der Taylorisierung verblüffte ja einst, wie Verfahrensweisen, die an der Fabrikarbeit entwickelt wurden, sukzessive das gesamte Wirtschaftsleben der industriellen Epoche erfassten – selbst vorindustrielle Sparten wie die Landwirtschaft und Bereiche, die das genaue Gegenteil der monotonen industriellen Arbeit darstellten, etwa künstlerische und wissenschaftliche Tätigkeiten. Insofern scheint es gut möglich, wenn auch heute im Detail schwer vorstellbar, dass Modelle, die in digitaler Wissensarbeit entwickelt werden, ebenfalls modifiziert zurückwirken auf ältere Bereiche von Industrie und Handel, Kultur und Kommunikation. Zweifelsfrei, das lässt sich in der globalen Diskussion verfolgen, kommt es lange schon zwischen jenen, die industriell denken und arbeiten, und jenen, die digital denken und arbeiten, zu einem Konflikt der Mentalitäten, der sich in seinen Qualitäten mit dem vergleichen lässt, der einst den Zusammenprall der alten agrarischen Mentalität und der neuen industriellen kennzeichnete.

Konkurrierende Kollaboration Mit der Professionalisierung der Open-Source-Praxis, die während der neunziger Jahre rund um Linux einsetzte, wurde es notwendig, den Wettbewerb um die begrenzte Zahl von Kunden auf eine andere als die übliche proprietäre Basis zu stellen. Der Lösungsansatz, der damals gefunden

> wurde, bestand in der Kombination von einerseits Kollaboration in Forschung und Entwicklung und andererseits Konkurrenz in Anwendung und Service. Wissen, das bekanntlich durch Teilung und Gebrauch weder weniger noch entwertet wird, wurde im globalen Datenraum gemeinsam produziert und geteilt, während man bei seiner Aufbereitung für Anwendung und Nutzung im Realraum verschiedene, lokalen und nationalen Bedürfnissen angepasste und durchaus auch konkurrierende Lösungen suchte. Zu den Wettbewerbsvorteilen, die diese neue Praxis dem Bruch mit den Traditionen des Geschäftslebens verdankt, zählen beschleunigtes Entwicklungstempo, gesteigerte Flexibilität in der Produktion sowie überlegene Qualitätskontrolle. Sie vor allem sorgt für die Konkurrenzfähigkeit der Open-Source-Produkte und -Dienstleistungen wie für deren zuverlässige Anpassung an spezifische Nutzerbedürfnisse.

In ihrer Gesamtwirkung erscheint die Open-Source-Praxis daher als wesentlicher Teil der kulturellen Adaptation an die kategorial neuen Möglichkeiten, die sich mit digitalen Technologien und der globalen Vernetzung eröffneten. Indem sie die Wissensarbeit auf ein neues Fundament stellte, ging ihr Siegeszug an die Wurzeln der industriellen Ordnung. Auffällig sind dabei die Parallelen zur Industrialisierung und insbesondere deren Frühzeit vor rund 200 Jahren. Denn auch damals erforderte eine neue – die industrielle – Technologie die Überwindung der etablierten Strukturen. Und auch damals standen im Zentrum der Entwicklung die beiden Elemente, um die heute wieder gerungen wird: die Verfügung über geistiges Eigentum und eine bessere, weil dem Potenzial der neuen Technologie entsprechende Arbeitsorganisation. Der Taylorismus allerdings, in dem diese epochale Anpassungsleistung um 1900 kulminierte, war mit der Unterwerfung des Einzelnen unter den industriellen Apparat nicht nur, wie oft bemerkt wurde, Ausdruck einer Maschinenwerdung des Menschen. Er war auch eine vordemokratische Organisationsform, in deren System der Entmündigung die faschistischen und kommunistischen Diktaturen bereits angelegt waren, die im 20. Jahrhundert in so vielen Ländern den Prozess der Industrialisierung vorantrieben.

Dagegen drücken sich in der Open-Source-Praxis gleichberechtigter und teilautonomer Kollaboration genuin demokratische Werte aus. Insofern steht die Open–Source-Praxis in grundsätzlichem Gegensatz zu dem System, das der Soziologe Manuel Castells als Industrialismus beschrieben hat.[29] In der Ausformung und Erprobung neuer Arbeits-, Verkehrs- und auch Rechtsformen, innovativer Managementverfahren, Eigentumsregelungen und Verhaltensweisen konturieren sich Strukturen einer neuen digitalen Ordnung, die nicht nur erhebliche Produktivitäts- und Wettbewerbsvorteile bietet, sondern auch Einzelnen ein größeres Maß an Freiheit einräumt, sie in höherem Maße motiviert und zu kreativer Partizipation befähigt.

Die Open-Source-Bewegung operierte so während der zweiten Hälfte des 20. Jahrhunderts an vorderster historischer Front. Wie Steven Johnson (1998) feststellte,

29 Vgl. Castells (2000).

brachte sie das zwangsläufig in Widerspruch zu traditionellen politischen Werten, die noch aus Zeiten des analogen Mangels an Ressourcen und eines erheblich langsameren Informationsflusses stammten:

> „Trying to talk about the politics of Open Source using the categories of traditional political taxonomy is like using mammalian anatomy to explain the respiratory apparatus of fish: there's a certain overlap, but the underlying architecture is so different that it's better off defining the terms from scratch".

Der damals oft recht unduldsame Enthusiasmus der Open-Source-Fans, der bei einer so prosaischen Angelegenheit wie der Software-Produktion befremdete, die bisweilen penetrante Mischung aus genialem Gestus, religiöser Erweckung und politischem Eifer, die sich bei einschlägigen Treffen beobachten ließ (und bisweilen noch lässt) – das alles wird unter dieser Perspektive zwar nicht erträglich, aber doch verständlicher: als Ausdruck des berechtigten Bewusstseins, Teil einer bahnbrechenden Avantgarde zu sein. Denn in der doppelten Anstrengung, die Verfügung der Menschheit über ihr Wissen und die Organisation der gesellschaftlichen Arbeit neu zu ordnen, war die Open-Source-Praxis um das Jahr 2000 in der Tat, wie Steven Johnson (1998) schreibt: „the digital worlds first truly indigenous political creation".

Die enge Verschränkung allerdings mit dem technischen Stand wie dem sozialen Gebrauch digitaler Vernetzung sorgte zur gleichen Zeit dafür, dass mit der zunehmenden Verfügbarkeit breitbandiger Immer-an-Vernetzung auch für kleinere Firmen und Privathaushalte sowie der Mobilisierung digitaler Vernetzung die Open-Source-Praxis den nächsten nachhaltigen Entwicklungsschub erhielt. Auf den Ursprung zur Zeit allein analoger Vernetzung und die Phase der Begründung zur Zeit des institutionellen Internets, die Open-Source-Praxis 1.0, sowie die Phase der Durchsetzung zur Zeit des privaten schmalbandigen Einwahl-WWW, die Open-Source-Praxis 2.0, folgte zu Beginn des 21. Jahrhunderts eine wiederum fortgeschrittenere Version 3.0.

Ihre Entstehung indizierte eine Vielzahl neuer Ansätze und Anstrengungen zur kollektiven Wissensproduktion, von emergierenden Systemen aggregierender Datenproduktion und -publikation – automatisierte Auswertung exemplarischer Konsumselektionen, nutzerproduzierte Reputations- und Empfehlungssysteme – bis hin zu spektakulären Web-2.0-Projekten wie *Wikipedia*, die mit Software-Architekturen der Partizipation zu Selbstausdruck und interaktiver Kollaboration in entorteter Echtzeit ermächtigen. Die akzelerierenden Vorgänge beobachtet und ihre Konsequenzen teils bedacht, teils imaginiert haben unter verschiedenen – medien- und kommunikationstheoretischen, psychologischen, systemtheoretischen, ökonomischen, juristischen – Perspektiven etwa Steven Johnson (2001), Howard Rheingold (2002), James Surowiecki (2004), Lawrence Lessig (2004) und Chris Anderson (2006).

Die ungemein spannende und viel versprechende Entwicklung führt freilich aus dem historischen Raum des 20. Jahrhunderts, den dieser Aufsatz untersuchen wollte, in

die unmittelbare Gegenwart. Beschreibung und Analyse der sich aktuell ausbildenden Open-Source-Praxis 3.0 müssen daher einer anderen Gelegenheit vorbehalten bleiben.

Literatur

Abbate, J. (1999), *Inventing the Internet*, MIT Press, Cambridge, MA, USA.

Anderson, C. (2006), *The Long Tail: Why the Future of Business is Selling Less of More*, 1. Aufl., Hyperion, New York.

Auletta, K. (2001), *World War 3.0, Microsoft and Its Enemies*, Random House, New York.

Battles, M. (2003), *Library : An Unquiet History*, W.W. Norton, New York.

Berners-Lee, T. (1989), 'Information Management: A Proposal'. http://www.w3.org/History/1989/proposal.html [05. Feb 2007].

Berners-Lee, T. und Fischetti, M. (1999), *Weaving the Web: The Past, Present and Future of the World Wide Web by Its Inventor*, Orion Business, London.

Bush, V. (1945), 'As We May Think', *The Atlantic Monthly* **176**(1), S. 101–108. http://www.theatlantic.com/doc/194507/bush [05. Feb 2007].

Campbell-Kelly, M. und Aspray, W. (1996), *Computer: A History of the Information Machine*, 1. Aufl., Basic Books, New York. The Sloan technology series.

Castells, M. (2000), *The Rise of the Network Society*, 2 Aufl., Blackwell Publishers, Malden MA.

Ceruzzi, P. E. (1998), *A History of Modern Computing*, MIT Press, Cambridge, MA, USA.

Charny, B. (2001), 'Microsoft Raps Open-Source Approach', CNET News.com. http://news.cnet.com/news/0-1003-202-5813446.html [05. Feb 2007].

Colwell, R. (1999), Major Leader Special Transcript: Rita Colwell, Director, National Science Foundation, Paper presented at the national press club newsmaker luncheon, National Press Club Washington D. C.

Cringely, R. X. (2002), 'Let's Get Small: Maybe the Best Place to Do Basic Research is in Bob's Garage', I, Cringely: The Pulpit. http://www.pbs.org/cringely/pulpit/2002/pulpit_20021031_000437.html [05. Feb 2007].

DiBona, C., Ockman, S. und Stone, M. (Hrsg.) (1999), *Open Sources: Voices from the Open Source Revolution*, 1. Aufl., O'Reilly, Cambridge, MA, USA. http://www.oreilly.com/catalog/opensources/book/toc.html [08. Dez 2006].

Engelbart, D. C. (1962), Augmenting Human Intellect: A Conceptual Framework, Summary Report, Stanford Research Institute. http://www.liquidinformation.org/ohs/62_paper_full.pdf [05. Feb 2007].

Foucault, M. (1971), *Die Ordnung Der Dinge*, 1. Aufl., Suhrkamp Verlag, Frankfurt am Main.

Foucault, M. (1988), *Archäologie Des Wissens*, 3. Aufl., Suhrkamp Verlag, Frankfurt am Main.

Frank, H. (1990), 'Oral history interview', Conducted by Judy O'Neill, Charles Babbage Institute, University of Minnesota. http://www.cbi.umn.edu/oh/pdf.phtml?id=123 [05. Feb 2007].

Free Software Foundation (2006), 'Die Definition Freier Software'. http://www.gnu.org/philosophy/free-sw.de.html [05. Feb 2007].

Freyermuth, G. S. (2000), 'Digitales Tempo', *c't – Magazin für Computertechnik* (14), S. 74–81.

Freyermuth, G. S. (2001*a*), 'Die neue Hackordnung. Aus der Open-Source-Geschichte lernen, Teil II', *c't – Magazin für Computertechnik* (21), S. 270. http://www.heise.de/ct/01/21/270/ [05. Feb 2007].

Freyermuth, G. S. (2001*b*), 'Es lebe das Chaos.', *Die Welt* **28. Apr 2001**. http://www.welt.de/data/2001/04/28/489144.html.

Freyermuth, G. S. (2001*c*), 'Offene Geheimnisse', *GDI Impuls* **2001**(2).

Freyermuth, G. S. (2001*d*), 'Offene Geheimnisse. Aus der Open-Source-Geschichte lernen, Teil I', *c't – Magazin für Computertechnik* (20), S. 176. http://www.heise.de/ct/01/20/176/ [05. Feb 2007].

Freyermuth, G. S. (2003), 'Die Geburt des PC aus dem Geiste des Protests: Eine kleine Kulturgeschichte', *c't – Magazin für Computertechnik* (24), S. 270.

Friedewald, M. (1999), *Der Computer als Werkzeug und Medium: Die geistigen und technischen Wurzeln des Personal Computers*, GNT-Verlag, Berlin.

Gantz, J. und Rochester, J. B. (2005), *Pirates of the Digital Millennium*, Prentice Hall/Financial Times, Upper Saddle River, NJ.

Gay, J. (Hrsg.) (2002), *Free Software, Free Society: Selected Essays of Richard M. Stallman*, GNU Press, Free Software Foundation, Boston, MA, USA. http://www.gnu.org/philosophy/fsfs/rms-essays.pdf.

Gillespie, T. (2007), *Wired Shut: Copyright and the Shape of Digital Culture*, MIT Press, Cambridge, MA, USA.

Gleick, J. (1999), *Faster: The Acceleration of Just About Everything*, Pantheon Books, New York.

Himanen, P. (2001), *The Hacker Ethic, and the Spirit of the New Economy*, 1. Aufl., Random House, New York.

Johnson, S. (1998), 'Open Source', Feed Magazine.

Johnson, S. (2001), *Emergence: The Connected Lives of Ants, Brains, Cities, and Software*, Scribner, New York.

Kanigel, R. (1997), *The One Best Way: Frederick Winslow Taylor and the Enigma of Efficiency*, Viking, New York. Sloan technology series.

Kern, S. (1983), *The Culture of Time and Space 1880-1918*, Harvard University Press, Cambridge, MA, USA.

Klotz, U. (1999), 'Neue Arbeitsplätze entstehen nicht durch mehr Verteilung. Neue ökonomische Spielregeln führen zu neuen Formen von Arbeit, Organisation und Politik', *Frankfurter Allgemeine Zeitung* **26. Jul 1999**.

Landes, D. S. (1969), *The Unbound Prometheus: Technological Change and Industrial Development in Western Europe from 1750 to the Present*, Cambridge University Press, London.

Landes, D. S. (1983), *Revolution in Time: Clocks and the Making of the Modern World*, Belknap Press of Harvard University Press, Cambridge, MA, USA.

Leonard, A. (1998), 'Let My Software Go!', *Salon online Magazine* **14. Apr 1998**. http://archive.salon.com/21st/feature/1998/04/cov_14feature.html [05. Feb 2007].

Lessig, L. (1999), *Code and Other Laws of Cyberspace*, Basic Books, New York.

Lessig, L. (2001), *The Future of Ideas: The Fate of the Commons in a Connected World*, 1. Aufl., Random House, New York.

Lessig, L. (2004), *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, Penguin Press, New York.

Levy, S. (1984), *Hackers: Heroes of the Computer Revolution*, Anchor Press/Doubleday, New York.

Levy, S. (2002), 'Lawrence Lessig's Supreme Showdown', *Wired* **10**(10). http://www.wired.com/wired/archive/10.10/lessig_pr.html.

Licklider, J. C. R. (1960), 'Man-Computer Symbiosis', *IRE Transactions on Human Factors in Electronics* **HFE-1**, S. 4–11. http://www.memex.org/licklider.pdf [05. Feb 2007].

Litman, J. (2006), *Digital Copyright*, Prometheus Books, Amherst, N.Y.

McHugh, J. (1998), 'For the Love of Hacking', *Forbes* **10. Aug 1998**. http://www.forbes.com/forbes/1998/0810/6203094a.html [05. Feb 2007].

Merton, R. K. (1973), *The Sociology of Science: Theoretical and Empirical Investigations*, University of Chicago Press, Chicago.

Mokyr, J. (1990), *The Lever of Riches: Technological Creativity and Economic Progress*, Oxford University Press, New York.

Moody, G. (2001), *Rebel Code: Inside Linux and the Open Source Revolution*, 1. Aufl., HarperCollings, New York.

Philipkoski, K. (2001), 'Biology Yearns to Be Free', Wired News. http://www.wired.com/news/medtech/0,1286,43151,00.html [05. Feb 2007].

Raymond, E. S. (1999), *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*, O'Reilly & Associates, Cambridge, MA, USA.

Rheingold, H. (2002), *Smart Mobs: The Next Social Revolution*, Perseus Pub., Cambridge, MA, USA.

Stone, A. R. (1995), *The War of Desire and Technology at the Close of the Mechanical Age*, MIT Press, Cambridge, MA, USA.

Surowiecki, J. (2004), *The Wisdom of Crowds: Why the Many are Smarter than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations*, 1. Aufl., Doubleday, New York.

Tapscott, D., Lowy, A. und Ticoll, D. (2000), *Digital Capital: Harnessing the Power of Business Webs*, Harvard Business School Press, Boston, MA, USA.

Tapscott, D. und Ticoll, D. (2000), 'Open Force: From Automobiles to Textbooks, the Linux Approach to Product Development Can Be a Powerful Tool for Every Business', *The Standard* **12. Jun 2000**.

Taylor, F. W. (1911), *The Principles of Scientific Management*, Harper & Brothers, New York, London.

The Economist (2001), 'Out in the Open', *The Economist* **12. Apr 2001**. http://www.economist.com/surveys/displayStory.cfm?Story_id=568269 [05. Feb 2007].

Tiemann, M. (2001), 'Decoding Microsoft's Open-Source Argument', CNET News.com. http://news.com.com/2010-1071-281468.html [05. Feb 2007].

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

Vaidhyanathan, S. (2001), *Copyrights and Copywrongs: The Rise of Intellectual Property and How It Threatens Creativity*, New York University Press, New York.

Waldrop, M. M. (2001), *The Dream Machine: J. C. Licklider and the Revolution That Made Computing Personal*, Viking, New York.

Wayner, P. (2000), *Free for All: How Linux and the Free Software Movement Undercut the High-Tech Titans*, 1. Aufl., Harper Business, New York.

Wells, H. G. (1937), 'World Brain: The Idea of a Permanent World Encyclopaedia', Contribution to the new Encyclopédie Française. http://sherlock.berkeley.edu/wells/world_brain.html [05. Feb 2007].

Wilcox, J. und Shankland, S. (2001), 'Why Microsoft is Wary of Open Source', CNET News.com. http://news.com.com/2100-1001-268520.html [05. Feb 2007].

von Hippel, E. (2000), 'Transfer the Job of Innovating to Users! An Interview with Eric von Hippel', *ZumThema* (41).

von Hippel, E. (2005), „Anwender-Innovationsnetzwerke": Hersteller entbehrlich, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 449–462. http://www.opensourcejahrbuch.de/download/jb2005/chapter_07/osjb2005-07-04-vonhippel [05. Feb 2007].

Die Deutungsoffenheit der Quelle

ALEXANDER KNORR*

(CC-Lizenz siehe Seite 563)

Jüngste ethnologische Arbeiten legen dar, dass Technologien keineswegs lediglich passiv konsumiert, sondern aktiv angeeignet werden. Kulturelle Aneignung bedeutet zunächst Umdeutung und Umwidmung der Technologien, ihre Einpassung in eigene soziale und kulturelle Muster, Vorstellungen und Praktiken. Doch das Spektrum der Aneignung reicht über die Neuzuschreibung von Bedeutung weit hinaus, bis hin zur Umarbeitung. Gerade bei letzterer sind gleichermaßen überraschende wie originelle handwerkliche und soziale Prozesse zu beobachten, die zu Erfindungen und Neuerungen führen. Eine entscheidende Rolle spielt dabei der Grad der Deutungsoffenheit[1] und der Umarbeitbarkeit des jeweiligen technologischen Artefakts. Im Falle von Software ist in diesem Zusammenhang die Zugänglichkeit des Quelltexts wesentlich. Ausgehend vom ethnologischen Konzept der soziokulturellen Aneignung wird die Bedeutung von Open Source für die Entstehung Innovationen hervorbringender kreativer Milieus gezeigt.

Schlüsselwörter: soziokulturelle Aneignung · soziokulturelle Vorstellungen · Kultur · soziokulturelle Praktiken

1 Einleitung

Sir Isaac Newton wird der Ausspruch zugeschrieben, wir stünden auf den Schultern von Giganten. Damit meinte er, dass jede Generation nur deshalb den wissenschaft-

* Mein besonderer Dank gilt Kurt Beck, dem jetzigen Lehrstuhlinhaber für Ethnologie an der Universität Bayreuth, der nicht nur die Ethnologie der Technik zeitgemäß revitalisiert, sondern mir mit „kultureller Aneignung“ das entscheidende Vehikel des Verständnisses vermittelt hat.

1 Ein Ding ist deutungsoffen, wenn seine Qualitäten es erlauben, damit auf verschiedene Arten und Weisen umzugehen. Dieser Umgang reicht von der Bedeutungszuschreibung über den Gebrauch bis zur Umarbeitung des Dinges. Je mehr Möglichkeiten des Umgangs mit dem Ding potenziell existieren, desto größer ist sein Grad der Deutungsoffenheit. Dabei ist zu beachten, dass der Grad der Deutungsoffenheit nicht zwingend mit der Komplexität des Dinges korrelliert.

lichen Horizont erweitern kann, weil die Summe der vor ihr erworbenen Erkenntnisse bereits eine hochgelegene Aussichtsplattform ermöglicht, auf der weiter aufgebaut werden kann. Falls Newton diesen Sinnspruch tatsächlich geäußert hat, so bezog er ihn wahrscheinlich auf die Naturwissenschaften, die Physik im Speziellen. Seine Metapher trifft aber nicht nur auf jede Art von Wissenschaft im abendländischen Sinne zu, sondern auch auf andere Bereiche, wie etwa Kunst oder Technologie. Wissen und Erkenntnis begnügen sich nicht damit, ein Dasein in Form von sprachlich kodierter Information zu fristen, sondern nehmen manifeste Gestalt an, werden zu materialisierter Kultur (Spittler 1993, S. 180), zu Artefakten – z. B. zu Maschinen oder Software. *Artefakt* ist ein sehr neutraler Begriff und bedeutet lediglich ein Ding, das künstlich geschaffen wurde, nicht naturgegeben ist und von einem denkenden und handlungsmächtigen Wesen in die Welt gebracht wurde. Aber in gängiger Diktion werden Maschinen und Software kaum als Artefakte, sondern als Produkte bezeichnet. Genau wie *Artefakt* nimmt auch der Begriff *Produkt* Bezug auf das Hergestelltsein, trägt aber weitere Implikationen in sich – namentlich die Unterscheidung und Trennung zweier Sphären: die der Hersteller und die der Kunden. Letztere werden gemeinhin als „(End-)Verbraucher" bezeichnet, im Zusammenhang mit Computern und Software oftmals als „Benutzer" (User). Diese Bezeichnungen wiederum implizieren das passive Konsumieren oder Gebrauchen eines Artefakts im Sinne seiner Produzenten. Doch diese Zuschreibungen sind fern der empirischen Wirklichkeit, denn mit erfolgter Produktion ist die Geschichte eines künstlich hergestellten Dinges keineswegs beendet (Kopytoff 1986).

Das hervorstechendste, ja vielleicht das wesentliche Merkmal des Menschen, das ihn von den anderen Tieren unterscheidet, ist seine Fähigkeit, in schwindelerregend rascher Abfolge Neuerungen in die Welt zu bringen. Diese Kreativität greift aber nicht nur auf Rohstoffe zurück, sondern auch auf Artefakte. Rohstoffe sind maximal formbar, im Grunde genommen setzen lediglich Vorstellungsgabe und Materialeigenschaften dieser Formbarkeit Grenzen. Aber auch Artefakte sind von einer interpretativen Flexibilität umgeben: Sie sind deutungsoffen (Beck 2001, S. 67).

> „The street finds its own uses for things—uses the manufacturers never imagined." (Gibson 1991, S. 27)

Das von dem Schriftsteller William Gibson so auf den Punkt gebrachte Phänomen wird in der zeitgenössischen Ethnologie mit den Theorien und Modellen der soziokulturellen Aneignung erfasst. Mit der Öffnung der Ethnologie zur Moderne und zur Realität des Globalen (Kearney 1995) rückte die materielle Kultur – d. h. Artefakte – wieder in den Fokus des Interesses (Spittler 1993). Industriell gefertigten, weltweit diffundierenden Waren und Gütern, und was mit diesen geschieht, wird seit einigen Jahren besondere Aufmerksamkeit geschenkt (Miller 1995). Die Beobachtung überraschender, nicht vorhersehbarer lokaler Umwidmungen, führte dazu, dass die Vorstellung von Anpassung (Bargatzky 1984; Beck 2000) an die kulturellen Vor-

stellungen der dominant erscheinenden stark industrialisierten Gesellschaften in den Hintergrund trat und durch eine Perspektive, welche die Handlungsmächtigkeit der Protagonisten am vermeintlich empfangenden Ende betont ersetzt wurde (Beck 1990, 2000; Spittler 1993, 2002). Aktives Handeln und Innovationen schaffende Kreativität sind die wesentlichen Aspekte, welche überraschende Ergebnisse ermöglichen, die nicht einfach Neuaufgegossenes alter Muster der beteiligten Gesellschaften sind. Das Paradigma der soziokulturellen Aneignung ist somit ein Gegenentwurf zu den Interpretationen von Globalisierung als entweder kulturell nivellierend oder kulturellem Fundamentalismus Vorschub leistend.

2 Aneignung

Das Spektrum dessen, was im neueren ethnologischen Sinne unter *Aneignung* verstanden wird, reicht vom Inbesitznehmen über Umdeutungen und Umwidmungen bis hin zur Umarbeitung. Diese Prozesse schweben weder im leeren Raum noch werden sie einfach nur von nach unseren Vorstellungen rationalen, pragmatischen oder opportunistischen Entscheidungen getragen, sondern hängen eng mit gesellschaftlichen Umständen und kulturellen Vorstellungen zusammen. Kulturelle Vorstellungen informieren die Aneignung, gesellschaftliche Gegebenheiten stecken den Rahmen der Möglichkeiten ab, der Prozess selbst und die resultierenden Artefakte wiederum entfalten Rückwirkung auf die aneignende Kultur und Gesellschaft und verändern diese. Soziokulturelle Aneignung ist somit ein dynamischer Prozess, der ineinander verschachtelte Rückkopplungsschleifen beherbergt.

2.1 Beschaffung

Jede Form von Aneignung beginnt damit, ein Ding in Besitz zu nehmen (Spittler 2002, S. 16 ff.). Bezüglich für den Markt gefertigter Güter ist uns der Prozess des käuflichen Erwerbens am vertrautesten, kommt uns als erstes in den Sinn, weil er unseren kulturellen Vorstellungen entspricht. In anderen Gesellschaften und Gruppen existieren andere Vorstellungen und damit auch andere, sozial akzeptierte Prozesse der Inbesitznahme. In vorkolonialer Zeit war bei den *Kel Ewey Tuareg* (Niger) der Raub eine sozial legitimierte Praxis. Im Zuge sogenannter Razzien oder Expeditionen, die unter großem Aufwand und Eingehen beträchtlicher Risiken in weit entfernte Gebiete führten, wurde Benötigtes wie Kamele, Nahrungsmittel, Sklaven und Güter gewaltsam beschafft (Spittler 2002, S. 21). Ein extremes Beispiel, zugegeben, aber gerade in Bezug auf die Thematik Software beileibe nicht an den Haaren herbeigezogen. Man denke an das Umgehen und Entfernen des Kopierschutzes kommerzieller Anwendungssoftware und Mediendateien (*cracken*) oder den gezielten Diebstahl von Dateien aus gesicherten Firmennetzwerken, der so etwas wie eine online durchgeführte Razzia im

Sinne der *Kel Ewey* darstellt.[2] Die spektakulärsten Fälle entstammen der Computerspieleszene, wie etwa der Diebstahl des Quelltextes von *Half Life 2* (*Valve Software*) oder einer Alpha-Version von *Doom III* (*id Software*), lange bevor die Spiele selbst auf den Markt kamen. *Rational choice*[3], gepaart mit rücksichtslosem Opportunismus zu kriminellem Handeln führend? Das ist eine natürlich nicht von der Hand zu weisende Interpretation, aber diese Perspektive verstellt einem den Blick auf die kulturellen und sozialen Dimensionen. Denn was passiert mit gecrackter Software, mit geraubten Dateien? Diejenigen, welche die Razzia durchgeführt haben, ganz gleich ob Individuum oder Gruppe, behalten die Beute nicht für sich, sondern verteilen sie, potenziell an jeden mit Internetzugang. Das wird erleichtert durch den Umstand, dass digitale Güter beliebig kopierbar sind (Kollock 1999), dennoch muss für die Verteilung, genauso wie vorher für die Expedition, beträchtlicher Aufwand getrieben werden. Ein durch gegenseitiges Vertrauen geprägtes Milieu muss ein Verteilungsnetzwerk schaffen, das die Sicherheit der Expeditionsmitglieder gewährleistet. Man sieht, die zeitlichen und sozialen Kosten sind hoch, doch wo ist der unmittelbare Nutzen? Fruchtbarer als die Suche nach Letzterem ist die Annahme, dass hier die von den Individuen eines Milieus geteilten Vorstellungen hinsichtlich des Zugangs zu und Besitzes von Software umgesetzt werden. In diesem Zusammenhang ist hinsichtlich der Verteilung das mittlerweile sehr populäre kollaborative File-Sharing-Protokoll *BitTorrent* (Cohen 2003)

2 Man könnte gegen diese Analogie einwenden, dass im Falle der westafrikanischen Razzien physische Gewalt zur Anwendung kommt, dass u. U. für das Überleben wesentliche Dinge geraubt werden oder sogar Menschen, was nach heute gängiger abendländischer Ethik den Grundsätzen des Menschseins widerspricht. Im Falle des Diebstahls von z. B. Software würden aber lediglich „immaterielle" Güter geraubt, was den Vergleich als unverhältnismäßig erscheinen lässt. So eine Beurteilung sieht aber die eigenen Überzeugungen als absolut gültig an; zur Interpretation und damit dem Verstehen der beobachteten Phänomene geeignet. Vernachlässigt wird hierbei der jeweils konkrete Kontext und die Innensicht der beteiligten Gruppen – beides ist für ein soziokulturelles Verständnis wesentlich. Wie *ernst* immaterielle Güter genommen werden, ist beispielsweise an den Reaktionen der filmschaffenden Industrie, v. a. in den USA, auf Brüche ihres Urheberrechts zu sehen (z. B. Verteilung von Spielfilmen via Internet): Es werden horrende Schadensersatzforderungen gestellt, die nicht nur dem außenstehenden Beobachter als jenseits jeglicher Verhältnismäßigkeit stehend erscheinen, sondern Individuen finanziell komplett ruinieren. In der Verfolgung und Prävention werden Maßnahmen ergriffen, die Bürgerrechte verletzen und die Exekutive zu verfassungswidrigem Handeln gebracht. Ein anderes Beispiel, das weiter unten noch angesprochen wird, ist die immer schärfere Handhabung des Patentrechts. In stark industrialisierten und hochgradig arbeitsteilig organisierten Gesellschaften können diese Praktiken Arbeitsplätze, ganze Firmen und damit de facto die Existenz Einzelner bedrohen. Wesentlich für ein Verstehen ist nicht die Beurteilung nach den Maßstäben des Beobachters, sondern das Erfassen der Sichtweisen der Beteiligten; wie z. B. im jeweiligen Kontext *Gewalt* oder *lebensbedrohlich* definiert wird, was sich aus Handlungen und Äußerungen ableiten lässt.

3 Die Rational-Choice-Theorien gehen von der Annahme aus, dass menschliches Handeln auf Zweckrationalität beruht (und damit eher Verhalten gleichkommt). Das heißt, dass menschliche Individuen stets diejenigen Mittel wählen, die ihnen zur Erreichung eines Bedürfnisses am optimalsten erscheinen. Dies impliziert eine stets waltende Nützlichkeitsmaximierung, und dass hinter jeder Handlung, die relevante Konsequenzen nach sich zieht, eine mehr oder weniger bewusste Kosten-Nutzen-Abwägung steht, wobei „Kosten" nicht nur als von monetärer Art gesehen werden, sondern gerade auch Zeitaufwand, soziale Kosten etc. berücksichtigt werden.

besonders interessant, weil das Prinzip des gegenseitigen Tausches (Reziprozität) bereits in das Protokoll und die Funktionsweise der Client-Software implementiert ist. Es sei betont, dass es hier nicht um eine Legitimation der erwähnten Praktiken geht, sondern um das Verstehen ihrer Hintergründe, die kollektiv geteilte Vorstellungen abgrenzbarer Gruppen sein können. Und selbstverständlich müssen auch offenkundig opportunistische Praktiken, wie das Einschleusen von *spyware* und *malware* [4] über besonders begehrte, illegal verteilte, gecrackte Software in Betracht gezogen werden.

2.2 Umdeutung

Ist ein Artefakt in einem Milieu angekommen, so kann es vorkommen, dass es den Intentionen der Hersteller entsprechend Verwendung findet. Sehr häufig jedoch wird es mit neuen Bedeutungen belegt und in den kulturellen Kontext eingebettet. Vor dem Hintergrund der Gegebenheiten ihres Lebensraumes erscheint es uns absolut vernünftig, dass die *Kel Ewey* häufig Sonnenbrillen tragen. Allerdings tragen sie die Brillen gar nicht in der Wüste, sondern bei Festlichkeiten, gerade auch nach Sonnenuntergang. Für die *Kel Ewey* komplettieren spiegelnde Sonnenbrillen die Verschleierung der Männer und perfektionieren damit die Umsetzung einer kulturellen Bekleidungsvorstellung (Spittler 2002, S. 18). Das Beispiel habe ich gewählt, weil einer Sonnenbrille zunächst eine eindeutige Funktion fest zugeordnet scheint, sie aber dennoch Umdeutungen erfährt, die einem selbst wohl kaum einfallen würden. Vergleichbares geschieht mit Software, die speziell für eindeutige Zwecke geschrieben wurde. Die Komplexität zeitgenössischer Computerspiele erlaubt es den Spielern, Entdeckungen zu machen und innerhalb des Spielraumes Dinge zu vollführen, die von den Entwicklern nicht beabsichtigt waren, ja häufig gar nicht für möglich gehalten, oder an die überhaupt nicht gedacht wurde. Speziell bei Multiplayer-Online-Spielen ermöglicht das Ausnutzen dieser Entdeckungen, zu betrügen und zu mogeln (*cheating*). Diese Praktiken werden von der herstellenden Industrie teilweise als Problem gesehen und es werden Maßnahmen ergriffen, um sie zu unterbinden (Zetterström 2001, S. 9–22). Seit dem Jahr 2000 entwickelt *EvenBalance Inc.* das Zusatzprogramm *PunkBuster* (mittlerweile in einer ganzen Reihe von Online-Spielen fest integriert), dessen einzig beabsichtigter Sinn es ist, *cheating* zu verhindern. Selbst bei längst etablierten Spielen muss *PunkBuster* ständig weiterentwickelt werden, weil immer neue Strategien des *cheating* auftauchen. Für die Szene der „professionellen *cheater*" besteht der Sinn des Computerspiels nicht mehr im Wettkampf mit anderen Spielern, sondern darin, die Gegenmaßnahmen immer wieder zu überlisten. Es hat sich ein Metaspiel entwickelt, was von den involvierten Akteuren auch genau so gesehen wird (Schwartzman

4 Als *spyware* wird Software bezeichnet, die Information aus einem Computersystem ohne Kenntnis des Besitzers an Dritte weitergibt. Unter *malware* versteht man Software, die ein Computersystem gegen den Willen seines Besitzers in seiner Funktion beeinträchtigt oder beschädigt. Sieht man die nicht beabsichtigte Weitergabe von Information aus dem eigenen System als Schaden an, wird *malware* zu einem Oberbegriff, unter den *spyware* fällt.

2001), das unvorhergesehene programmiertechnische Innovationen zeitigt. Auch die Praktiker des *game modding* haben den Spielen eine neue Bedeutung zugeschrieben. *Game modding* bedeutet das Herstellen spielbarer Zusätze zu kommerzieller Computerspiele-Software bis hin zur Entwicklung vollkommen neuer Spiele, die mit dem Ursprungsspiel nur noch dessen Basis, die *game engine* gemein haben. Ein Informant aus der Online-Gemeinschaft, bei welcher ich meine Feldforschung durchführte, hat diese neue Bedeutung auf den Punkt gebracht:

„I hardly play those games, I immediately start to mod them."

Die unkontrollierbare Kreativität der Spielergemeinschaft transformiert den Konsumgegenstand Computerspiel in ein co-kreatives Medium (Kuecklich 2004, S. 7), was uns zur stärksten Form der Aneignung, zur Umarbeitung, führt.

2.3 Umarbeitung

Aneignung bedeutet nicht nur symbolische Umdeutung, sondern auch kulturell angeleitetes Handeln (Spittler 1993), aktives In-die-Hand-Nehmen und tatsächliches Verändern von Artefakten (Beck 2001, 2003, 2004). In Bezug auf Maschinen und Technologie sind die Freiheitsgrade der Aneignung zu einem gewissen Grad eingeschränkt, wenn der Wunsch besteht, dass die Maschine nach der Aneignung immer noch funktionieren soll (Beck 2003). Dieser Umstand führt häufig zu dem Fehlschluss, der von den Ingenieuren eingeschriebene Sinn einer Maschine sei dominant, verhindere eine tiefgreifende Aneignung jenseits des Oberflächlich-Optischen, des Symbolisch-Ästhetischen. Mittlerweile sind uns die Bilder der prunkvollen, mit Verzierungen überladenen Busse und LKWs aus z. B. Indien und Pakistan vertraut geworden. Die auf sie getürmten Götter- und Heiligenbilder, Kalligraphie und exotisch wirkenden Muster zeigen uns, dass diese Maschinen der Ästhetik einer uns zunächst fremden Kultur unterworfen wurden. Man staunt und lächelt, etwa so wie über den längst sprichwörtlich gewordenen, getunten *Opel Manta*. Das Blenden der bizarren Schönheit verstellt den Blick auf die viel weiter reichenden, für das Verständnis der Beziehungen zwischen Technik und Kultur bedeutsamen Aneignungsprozesse. Im Sudan fahren *Sifinja* (Sandale) genannte LKWs, die auf ihre Art auch imposante Schönheiten sind, doch vom Äußeren her bei Weitem nicht so auffallen, wie ihre asiatischen Cousins – die erstaunlichen Modifikationen liegen hier etwas im Verborgenen. Bei tiefergehender Betrachtung stellt man fest, dass das ursprüngliche Fahrzeugmodell, der *Bedford TJ*, nur als Plattform für die weitreichenden Umbauten gedient hat, dass von ihm nahezu nichts mehr übrig ist. Die Ladefläche und das Fahrgestell wurden verlängert, die Längsträger auf doppelte Stärke gebracht, die Querträger verkürzt. Letztere werden durch zusätzliche Stahlstangen entlastet, zusätzliche Federblätter sowie verlängerte Stoßdämpfer und Umlenkhebel wurden eingesetzt. Ein völlig neu konzipierter Kabelbaum wurde eingezogen, der Tank ist verstärkt und vergrößert, genau wie die Räder auf der Antriebsachse. Der Auspuff wurde komplett mit Krümmer

und Schalldämpfer von den lokalen spezialisierten LKW-Schmieden hergestellt. Fahrerkabine und Motorverkleidung sind vom restlichen Fahrzeug abgetrennt und auf einer ureigenen Erfindung der sudanesischen Schmiede schwimmend gelagert und somit von Verwindungen entlastet. Jetzt trägt der LKW acht statt sechs Tonnen, ist wendiger und meistert die extremen Verhältnisse der Querfeldeinpisten, ohne von den auftretenden Kräften über kurz oder lang zerrissen zu werden. Außer den erwähnten Schmieden sind am Umbau Polsterer, Elektriker, Lackierer, Schreiner, Mechaniker, Kalligraphen, die Fahrer und deren Assistenten beteiligt. Wissen und Erfahrung all dieser Gruppen fließen zu einer kollektiven Vision eines besseren LKWs zusammen, wie sie sich schließlich in der *Sifinja* manifestiert (Beck 2004, S. 252–258). Betont werden muss, dass es sich hier nicht um eine unorganisierte Bande dilettierender Amateure und Hobbyisten handelt, die aus der Not heraus mehr schlecht als recht an schrottreifen LKWs herumbasteln und -pfuschen. Vielmehr werden von einem professionellen und selbstbewussten Milieu voll funktionierende LKWs bis auf ihr Skelett dekonstruiert und neu erschaffen. Ein Teil des Stolzes des Milieus liegt darin, brachliegende Potenziale des *TJ* so zur Blüte zu bringen, wie es die Ingenieure bei *Bedford* nie gewagt hätten. Dies ist aber nur möglich, weil der *TJ* zugänglich verschraubt ist und weder versiegelte Komponenten noch komplizierte Elektronik besitzt (Beck 2004, S. 251). Metaphorisch gesprochen: Der Quelltext des Systems liegt offen. Entsprechend unterscheidet sich die Alpha-Version von *Doom III*, die durch das Internet sickerte, von der später legal vertriebenen Version unter anderem dadurch, dass weite Teile des Programms viel zugänglicher sind. Innerhalb von Tagen, nachdem er die Version in die Hände bekam, hatte mein bereits erwähnter Freund und Informant nicht nur die Third-Person-Perspektive für den Spieler ermöglicht, sondern auch den sogenannten Bullet-Time-Effekt[5] verwirklicht, was von den Autoren der *game engine* nie vorgesehen war. Der Grad der Deutungsoffenheit der Alpha-Version erlaubte die Umsetzung von Visionen eines besseren Spiels; eines Spiels, das im Milieu der *game modder* zirkulierenden Vorstellungen mehr entsprach. Genauso wie GNU/Linux den der Open-Source-Gemeinschaft eigenen Vorstellungen von einem Betriebssystem entspricht.

2.4 Vergesellschaftung

Jedes kreative Individuum steht nicht nur potenziell auf den Schultern von Giganten, sondern ist auch an der Hand genommen, kann sich anlehnen und aufstützen auf seine Zeitgenossen, die Menschen um ihn herum, mit denen es kommuniziert und

5 Das Konzept *bullet time* geht auf die Arbeiten von Dayton Taylor zurück (Knorr 2000), erfuhr durch die *Matrix*-Filme immense Popularität und wurde mit dem Computerspiel *Max Payne* (*Remedy Entertainment*) erstmals interaktiv erfahrbar. Der Begriff selbst wurde von *Remedy Entertainment* geprägt und später an *Warner* verkauft. *Bullet time* bedeutet extreme Zeitlupe bis hin zum Einfrieren einer Szenerie, wobei die Kamera aber nach wie vor im Raum bewegt werden kann. Als dramatisches Element in Computerspielen erlaubt es dem Spieler, in Echtzeit zu agieren, während alle anderen Vorgänge im Spielraum extrem verlangsamt ablaufen.

interagiert. Selten, wenn überhaupt, werden Neuerungen von Einhandseglern und Einzelgängern ersonnen und umgesetzt. Im Regelfall werden Arbeitsteilung und Austausch genutzt. Letztere sind nicht nur in bereits bestehende soziokulturelle Muster eingebettet, sondern gehen auch keineswegs chaotisch und planlos vor sich, folgen vielmehr Regeln und Mustern – neue Formen sozialer Organisation, Struktur (Lehmann 2004) und Tradition scheinen auf. Aus den Dorfschmieden von Shendi, El Obeid, Port Sudan und Hillat Kuku (Khartum-Nord) haben sich in nur wenigen Generationen Milieus von LKW-Schmieden entwickelt, deren Schmiedetradition aber immer noch unverkennbar ist. Die verschiedenen neuen Traditionen sind nicht gegenseitig voneinander abgegrenzt, sondern befruchten sich flexibel wechselseitig. Sie sind nicht eingeschränkt durch staatliche Überwachung, erstarrte Sozialstruktur und kulturelle Traditionen des Handwerks, sondern frei von standardisierten Entwicklungs- und Produktionsabläufen des euroamerikanischen und japanischen Ingenieurwesens (Beck 2004, S. 260). Entsprechende in Freiräumen existierende kreative Milieus kennzeichnen wesentlich die gesamte Geschichte der Computertechnologie (Levy 1984).

Bis zum Ende der 1950er Jahre enthoben stark ritualisierte Praktiken des Umgangs mit Computern – sowie eine entsprechende Architektur – diese Maschinen dem Reich des Profanen und verliehen ihnen eine Aura von Sakralem (vgl. Katz 1981). Die damals noch riesenhaften Rechner standen unzugänglich in speziell gebauten Räumlichkeiten, nicht einmal den Verfassern von Programmen war direkter Zugriff erlaubt. Die Programmierer mussten mit Stapeln von Lochkarten an Schaltern anstehen, wo ihre Programme von den hohen Priestern, angemessen gewandet in langen weißen Labormänteln, entgegengenommen wurden, um von den Blicken entzogen im verbotenen Inneren der Kathedrale schließlich eingelesen zu werden. Die Ergebnisse wurden den Programmierern zugestellt. Bestand der Rücklauf nur aus Fehlermeldungen, lief das gesamte Procedere von neuem an. Diese Verfahrensweise gebar sicherlich Ehrfurcht, aber auch lange Entwicklungszeiten der Software. Mit dem Beginn der 1960er Jahre und dem Einzug einer neuen Generation von Computern änderten sich die Strukturen, die Priester verschwanden, direkter Zugang und damit echtes interaktives Programmieren wurde möglich, wodurch die Produktionszeit wunschgemäß laufender Software signifikant verringert werden konnte, da jetzt bei Fehlern die Programmierer unmittelbar Rückmeldung von der Maschine erhielten. Dafür tauchte für ähnliche Aufgaben und Probleme eine breite Palette verschiedener Lösungen auf, die sich hinsichtlich Effizienz und Eleganz erheblich voneinander unterschieden.

Mit den Priestern waren auch die Schlangen von Programmierern vor den Schaltern verschwunden. Ein informelles Milieu des Austauschs hatte aufgehört zu existieren (Graetz 1981; Lévy 2002; Weinberg 1988). Durch von kulturellen Vorstellungen geprägte soziale Strukturen wurde zwar der Zugang zur Rechenleistung eingeschränkt, aber auch hier entwickelten sich Kreativität und Innovationen hervorbringende Nischen freien, ungezwungenen Austauschs. Um Eric Raymonds (1999) Metapher auf-

zugreifen: Nicht nur im Neuen Testament, sondern auch im damaligen Umfeld der gigantischen *Mainframes* finden wir in der Kathedrale einen Basar. Der Einsturz und das Verschwinden der Kathedralen riss auch die in sie eingebetteten Nischen mit sich, aber das Fallen der Restriktionen löste umgehend die Entstehung eines für die Entwicklung der Computertechnologie zentralen informellen Milieus aus. In den freien akademischen Umgebungen der Universitäten blühte die Hackerkultur auf, die mit Fug und Recht als Kultur bezeichnet werden kann, wie die in den verschiedenen Hackerethiken ausgedrückten kollektiv geteilten Werte, Normen, Vorstellungen und Ideen belegen (Levy 1984). Handeln und Praktiken der frühen Hacker deckten die gesamte Bandbreite der kulturell informierten Aneignung des Computers ab und zeitigten überraschende Ergebnisse, die im weiteren Verlauf der Geschichte ungeahnte Wirkungen entfalten sollten. Natürlich hatten die Universitätsverantwortlichen bestimmte Vorstellungen davon, wie und für was die immer noch sehr teuren Maschinen verwendet werden sollten. Der *PDP-1* etwa, der im Jahr 1961 immer noch ungefähr 120 000 US-Dollar kostete, war eindeutig als Werkzeug für wissenschaftliche Zwecke gedacht. Doch eine Gruppe von MIT-Studenten reinterpretierte ihn als Spielmaschine und schrieb *Spacewar*, eines der allerersten Computerspiele (Graetz 1981).

> „Objekte und Begriffe ändern mit dem Gebrauch, den verschiedene Akteure von ihnen machen, auch ihren Sinn." (Lévy 2002, S. 921–922)

Niemand konnte voraussehen, dass aus dieser Umdeutung eine bedeutende Industrie entstehen würde,welche heute die wesentliche sozioökonomische Triebfeder für die Weiterentwicklung von Computerhardware darstellt (Montfort 2002). Notwendige Vorbedingung für die Erfindung des Computerspiels war die Deutungsoffenheit des Systems *Computer*, die nicht durch soziale oder technische Restriktionen eingeschränkt wurde. Die Universitäten gewährten freien Zugang zur Hardware – Software war damals ohnehin frei, denn der Computermarkt bezog sich ausschließlich auf Hardware (Grassmuck 2000, S. 5). Erst mit den 1980er Jahren setzte die Kommodifizierung von Software ein, der Quelltext von sogenannter proprietärer Software wurde unzugänglich gehalten und letztere büßte entscheidend an Deutungsoffenheit ein. Doch die Werte der seit 20 Jahren existierenden Hackerkultur brachen sich eine neue Bahn und manifestierten sich in der Gegenbewegung GNU/Linux, woraus das, was wir heute als Open-Source-Bewegung kennen, entstand.

Die informellen Milieus des Basars sind die Brutstätten der Kreativität (Beck 2004), die im Falle von Open-Source-Projekten zumeist in den durch die Internetinfrastruktur aufgespannten konzeptuellen Räumen existieren. Der Umstand, dass das geteilte Kerninteresse, um welches Open-Source-Gemeinschaften kondensieren, genau im Bereich der Technologien liegt, welche die Interaktion ermöglichen, versetzt diese Gruppen in die Lage, sich die Internetinfrastruktur selbst anzueignen. Die hohe Vertrautheit mit dem Protokollstapel des Internets und allen weiteren involvierten

Technologien erlaubt es ihnen, sich virtuos Interaktionsräume zu schaffen, die nicht nur effiziente Kollaboration ermöglichen, sondern auch eine große Bandbreite sozialen Handelns und kulturellen Ausdrucks (Knorr 2006*a*, S. 6). Der geographischen Verteiltheit der Mitglieder solcher Gruppen zum Trotz führt dies zur Entstehung *echter* Gemeinschaften im sozialwissenschaftlichen Sinn. Es ist widersinnig, hier von *virtuellen Gemeinschaften* zu sprechen, weil die einzelnen Individuen nicht scheinbar interagieren, sondern tatsächlich (Knorr 2006*b*). Die Vorstellung, dass eine soziale Körperschaft ausschließlich durch über Technologien vermittelte Interaktion aufrechterhalten werden kann, ist nach wie vor für viele unvertraut, was ein erstes Hindernis für das Verstehen des Open-Source-Phänomens darstellt, da die wesentliche Interaktion zwischen den beteiligten Akteuren fast ausschließlich online geschieht. Genau hier können die spezifischen Potenziale der Ethnologie ihre Wirkung entfalten, denn heutige ethnologische Begriffe, Theorien und Methoden sind in einem Prozess entstanden, der stets in einem Kontext der Auseinandersetzung mit zunächst vollkommen unverstandenen, fremdkulturellen Wirklichkeiten stand und steht. Somit neigt ein ethnologischer Ansatz weniger dazu, von eurozentristisch gefärbten, unausgesprochenen Vorannahmen in fruchtlose Richtungen geführt zu werden. Dies gilt nicht nur hinsichtlich der Akzeptanz von vermittelter Interaktion als legitime Grundlage des Kulturellen und Sozialen von „lediglich" online wahrnehmbaren Gemeinschaften, sondern auch für die soziokulturellen Prinzipien hinter ihrem Funktionieren. In Bezug auf Open Source ist das originär aus ethnologischer Forschung stammende Paradigma der Gabenökonomie[6] zu einem echten Exportschlager der Ethnologie geworden, dass von allen sozialwissenschaftlichen Disziplinen, die sich mit dem Phänomen beschäftigen, importiert und erkenntnisbringend angewendet wurde (siehe z. B. Barbrook 1998; Bergquist und Ljungberg 2001; Kelty 2001, 2004, 2005).

6 Bereits in der ersten Hälfte des 20. Jahrhunderts stellten Ethnologen fest, dass in vielen außereuropäischen Gesellschaften der Fluss und die Zirkulation von Gütern mit Modellen der Marktwirtschaft nicht zu greifen und zu verstehen ist. Mauss (1990) rezipierte diese ethnologischen Befunde und baute darauf das Modell der Gabenökonomie auf. Wesentlich dabei ist, dass die weitergegebenen Güter nicht als Waren (mit einem Preis versehene und auf dem Markt handelbare Güter) gesehen werden, sondern als Gaben (mit Prestige, Status, Erwartungen, Rechten oder Pflichten „aufgeladene" Güter). Beispiele von Gaben in der deutschen Gesellschaft sind z. B. das Weihnachtsgeschenk oder Dienstleistungen im Rahmen nachbarschaftlicher Hilfe. In diesen Fällen wird nicht erwartet, dass die Gabe mit Geld bezahlt, sondern durch eine Gegengabe erwidert wird – beides kann nicht im Sinne von Preis oder Börsenkurs notiert werden, sondern entspricht sozialen Verhältnissen und kulturellen Erwartungen, etwa eine Einladung zum Abendessen für das Ausleihen eines Rasenmähers etc. Wichtige Prinzipien der Zirkulation von Gütern in Gabenökonomien sind die Gegenseitigkeit (Reziprozität) und die Umverteilung (Redistribution). Mauss' Werk fiel, und fällt bis heute, in vielen akademischen Disziplinen auf fruchtbaren Boden. Das von ihm formulierte Modell wurde vielfach weiterentwickelt und ermöglichte ein besseres Verständnis von Phänomenen, bei denen man mit „gängigen" Modellen von Ökonomie nicht weiterkam und die sich beileibe nicht auf außereuropäische Gesellschaften beschränken.

3 Folgerung

Die in der Einleitung aufgestellte Behauptung, lediglich Vorstellungsgabe und Materialeigenschaften würden der Formbarkeit von Rohstoffen Grenzen setzen, ist mit dem Zusatz „im Grunde genommen" relativiert, denn Ressourcenverteilung und Machtstrukturen wurden nicht erwähnt. Auch in Bezug auf die Aneignung von Artefakten sind diese Aspekte nicht zu vernachlässigen. Ja, der Quelltext ist rohstoffähnlich, wenn man die entsprechende Programmiersprache beherrscht, aber damit sind beileibe nicht die notwendigen Voraussetzungen erfüllt. Die Nilfellachen des Sudan können den Dieselmotor öffnen, auseinandernehmen und verändern (Beck 2001), die Schmiede von Shendi können den *Bedford TJ* zerlegen, modifizieren und völlig neu zusammensetzen (Beck 2004), aber weder die einen noch die anderen können einen Motorblock gießen, geschweige denn – noch basaler – Erz zu dem benötigten Eisen und Stahl verhütten. Genauso wenig kann ein Open-Source-Programmierer einen Mikrochip herstellen oder die Infrastruktur des Internets aufrechterhalten. Dazu sind Ressourcen von einer Größenordnung notwendig, zu deren Verwaltung nur die privatwirtschaftliche Industrie in der Lage ist.

> „Die Metamorphose ist [. . .] alles andere als voraussetzungslos. Damit stellt sich die Frage nach der Kreativität. Ich denke, die Kreativität der Aneignung gedeiht vor allem unter Bedingungen eines relativen Wohlstands in der Armut, hauptsächlich jedoch Professionalität, Nähe zu den zu lösenden Aufgaben und handwerklicher Unabhängigkeit, wozu nicht nur unternehmerische Selbständigkeit, sondern auch eine handwerkliche Praxis gehört, die sich kulturelle Autonomie gegenüber der europäischen Ingenieurstradition bewahrt hat." (Beck 2004, S. 257–258)

Open Source ist sicherlich keine Kultur der Armut, aber doch zumindest eine Kultur des Budgets, eingebettet in ein größeres Ganzes. Dessen sind sich die Akteure sehr bewusst. Open Source bedeutet keine Revolution (DiBona et al. 1999), sondern Widerstand. Trotz mit voller Absicht extrem und provokant formulierter Aussagen, wie der sehr bekannten von Richard Stallman, Software sei ein Menschenrecht, lässt sich bei Open-Source-Praktikern keineswegs eine kollektiv geteilte und dominante Vorstellung feststellen, die besagt, das bestehende ökonomische System der globalen Marktwirtschaft müsse beseitigt und durch ein völlig anderes ersetzt werden. Vielmehr dreht es sich um die Umsetzung von Open-Source-Praktiken und -Vorstellungen in gangbare Geschäftsmodelle, die grundsätzlich in das gegenwärtig dominante Modell von Marktwirtschaft eingepasst werden können und sollen. Es wird kein kultureller Fundamentalismus betrieben, auch wenn die Schriften und Auftritte der prominenten Open-Source-Evangelisten wie Bruce Perens und Richard Stallman diesen Eindruck erwecken können. Die weitreichenden Bemühungen des Milieus, entsprechende justiziable Lizenzen zu schaffen, legen davon Zeugnis ab. Die Lizenzen selbst sowie der

sich um sie rankende Diskurs zeigen, dass hier kein Umsturz zu völlig anderen Modellen von Ökonomie und Gesellschaft, etwa gar sozialistischer oder kommunistischer Prägung, vorangetrieben wird. Stattdessen werden auf soziokulturellen Vorstellungen und Werten der Open-Source-Bewegung basierende Produktions-, Kooperations- und Verteilungsmodelle (Gläser 2003) in Termini der Prinzipien geltenden internationalen Rechts gefasst, um eine juristisch abgesicherte Einpassung zu ermöglichen. Genau wie technische Artefakte und soziale Strukturen, sind auch ökonomisch-juristische Systeme deutungsoffen, können verändert und erweitert werden.

Die ebenfalls in der Einleitung vorgenommene Analyse der Implikationen von Begriffen wie *Produkt* und *Konsument* ist keine rein semantische, schöngeistige Interpretation, sondern weist auf in bestimmten Milieus existierende Haltungen hin, die in entsprechenden, empirisch wahrnehmbaren Praktiken ihren Ausdruck finden. Die restriktive Handhabung des Urheberrechts, das Versiegeln von Maschinenkomponenten, das Notwendigmachen von Spezialwerkzeugen und das Zurückhalten des Quelltexts verringern die Deutungsoffenheit von Artefakten und drücken damit Kreativität und Innovationen bis zu einem gewissen Grad die Luft ab. Man kann hier aber nicht einfach einer Seite den „Schwarzen Peter" zuschieben, denn hier stehen sich schlichtweg unterschiedliche kulturelle Vorstellungen gegenüber. Auch Lawrence Lessig weist darauf hin, wenn er schreibt, dass „die Industrie" nicht „böse" sei, sondern die Gesellschaft als Ganzes in der Verantwortung stehe (Lessig 2004, S. 260). Genau diese Verantwortung kann aber erst dann wahrgenommen werden, wenn die kulturellen Aspekte und Hintergründe der verschiedenen beteiligten Gruppen verstanden sind. Kultur- und Sozialwissenschaften wie die Soziologie, aber eben auch die Ethnologie besitzen dazu entsprechende Methoden, Konzepte und Modelle, wie die für ein Verstehen des Open-Source-Phänomens fruchtbar angewendete Theorie der Gabenökonomie bereits gezeigt hat.

Literatur

Barbrook, R. (1998), 'The High Tech Gift Economy', *First Monday* **3**(12). http://www.firstmonday.org/issues/issue3_12/barbrook/ [08. Dez 2006].

Bargatzky, T. (1984), 'Culture, Environment, and the Ills of Adaptationism', *Current Anthropology* **25**(4), S. 399–415.

Beck, K. (1990), 'Entwicklungshilfe als Beute: Über die lokale Aneignungsweise von Entwicklungsmaßnahmen im Sudan', *Orient: Deutsche Zeitschrift für Politik und Wirtschaft des Orients* **31**(4), S. 583–601.

Beck, K. (2000), Aneignung – Jenseits der Akzeptanzforschung, *in* R. Kappel et al. (Hrsg.), 'Afrika 2000 – 17. Tagung', Leipzig. CD-ROM.

Beck, K. (2001), Die Aneignung der Maschine, *in* K.-H. Kohl und N. Schaffhausen (Hrsg.), 'New Heimat', Lukas & Sternberg, New York, S. 66–77. Katalog zur Ausstellung im Frankfurter Kunstverein.

Beck, K. (2003), Die gestohlene Bank, Vortrag vom 7. Mai im Rahmen des Kolloquiums Ethnologische Theorien, Institut für Ethnologie und Afrikanistik, Ludwig-Maximilians-Universität, München.

Beck, K. (2004), Bedfords Metamorphose, *in* K. Beck, T. Förster und H. P. Hahn (Hrsg.), 'Blick nach vorn: Festgabe für Gerd Spittler zum 65. Geburtstag', Rüdiger Köppe, Köln, S. 250–263.

Bergquist, M. und Ljungberg, J. (2001), 'The Power of Gifts: Organizing Social Relationships in Open Source Communities', *Information Systems Journal* **11**(4), S. 305–320.

Cohen, B. (2003), 'Incentives Build Robustness in BitTorrent', Workshop on Economics of Peer-to-Peer Systems. http://bitconjurer.org/BitTorrent/bittorrentecon.pdf [02. Dez 2006].

DiBona, C., Ockman, S. und Stone, M. (Hrsg.) (1999), *Open Sources: Voices from the Open Source Revolution*, 1. Aufl., O'Reilly, Cambridge, MA, USA. http://www.oreilly.com/catalog/opensources/book/toc.html [08. Dez 2006].

Gibson, W. F. (1991), Academy Leader, *in* M. Benedikt (Hrsg.), 'Cyberspace: First Steps', MIT Press, Cambridge, London, S. 27–29. http://cyberpunk.barzha.com/f/gibs15.html [08. Dez 2006].

Gläser, J. (2003), A highly efficient Waste of Effort: Open Source Software Development as a specific System of collective Production, *in* 'Proceedings of the TASA 2003 Conference', University of New England. http://repp.anu.edu.au/GlaeserTASA.pdf [08. Dez 2006].

Graetz, M. J. (1981), 'The Origin of Spacewar', *Creative Computing* **7**(8), S. 56–67.

Grassmuck, V. (2000), Freie Software: Geschichte, Dynamiken und gesellschaftliche Bezüge, Abschlußbericht des DFG-Forschungsprojektes *Von der Ordnung des Wissens zur Wissensordnung digitaler Medien*, Humboldt-Universität zu Berlin. http://mikro.org/Events/OS/text/freie-sw.pdf [04. Dez 2006].

Katz, P. S. (1981), 'Ritual in the Operating Room', *Ethnology* **20**(4), S. 335–350.

Kearney, M. (1995), 'The Local and the Global: The Anthropology of Globalization and Transnationalism', *Annual Review of Anthropology* **24**, S. 547–565.

Kelty, C. M. (2001), 'Free Software / Free Science', *First Monday* **6**(12). http://www.firstmonday.org/issues/issue6_12/kelty/index.html [08. Dez 2006].

Kelty, C. M. (2004), 'Culture's Open Sources', *Anthropological Quarterly* **77**(3). http://aq.gwu.edu/~gwaq/aq_cultures_opensources.pdf [08. Dez 2006].

Kelty, C. M. (2005), 'Geeks, Social Imaginaries, and Recursive Publics', *Cultural Anthropology* **20**(2), S. 185–214.

Knorr, A. (2000), 'Matrix – Rezension der Wirklichkeit (1): Die Magier des Lichts', *Magische Welt* **49**(2), S. 169–174.

Knorr, A. (2006*a*), The Online Nomads of Cyberia, Presentation, 9th EASA Biennial Conference. http://www.media-anthropology.net/knorr_online_nomads.pdf [02. Dez 2006].

Knorr, A. (2006*b*), The Stability of Cyberspace, *in* R. Polcak (Hrsg.), 'Proceedings of the Cyberspace 2005 International Conference'. http://www.ethnologie.lmu.de/downloads/KNORR_2006_The_stability_of_cyberspace.pdf [02. Dez 2006].

Kollock, P. (1999), The Economies of Online Cooperation: Gifts and Public Goods in Cyberspace, *in* M. Smith und P. Kollock (Hrsg.), 'Communities in cyberspace', Routledge, London, New York, S. 220–242.

Kopytoff, I. (1986), The Cultural Biography of Things: Commoditization as Process, *in* A. Appadurai (Hrsg.), 'The Social Life of Things: Commodities in Cultural Perspective', Cambridge University Press, Cambridge, S. 64–91.

Kuecklich, J. (2004), Other Playings – Cheating in Computer Games, *in* 'Proceedings of the Other Players Conference', Kopenhagen. http://itu.dk/op/papers/kuecklich.pdf [02. Dez 2006].

Lehmann, F. (2004), 'FLOSS Developers as a Social Formation', *First Monday* **9**(11). http://www.firstmonday.org/issues/issue9_11/lehmann/index.html [08. Dez 2006].

Lessig, L. (2004), *Free Culture – How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, Penguin Press, New York, USA.

Levy, S. (1984), *Hackers: Heroes of the Computer Revolution*, Anchor Press/Doubleday, New York.

Lévy, P. (2002), Die Erfindung des Computers, *in* M. Serres (Hrsg.), 'Elemente einer Geschichte der Wissenschaften', Suhrkamp, Frankfurt am Main, S. 905–944.

Mauss, M. (1990), *Die Gabe*, Suhrkamp Taschenbuch Wissenschaft, Frankfut am Main.

Miller, D. (1995), 'Consumption and commodities', *Annual Review of Anthropology* **24**, S. 141–161.

Montfort, N. (2002), 'From PlayStation to PC', *Technology Review* **105**(2), S. 68–73.

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

Schwartzman, C. (2001), 'Interview with a Cheater'. http://www.3dactionplanet.com/features/interviews/xqz2/ [02. Dez 2006].

Spittler, G. (1993), 'Materielle Kultur – Plädoyer für eine Handlungsperspektive', *Zeitschrift für Ethnologie* **118**, S. 178–181.

Spittler, G. (2002), Globale Waren – Lokale Aneignungen, *in* B. Hauser-Schäublin und U. Braukämper (Hrsg.), 'Ethnologie und Globalisierung: Perspektiven kultureller Verflechtungen', Reimer, Berlin, S. 15–30.

Weinberg, G. M. (1988), *The Psychology of Computer Programming*, Van Nostrand Reinhold Co., New York.

Zetterström, J. (2001), A Legal Analysis of Cheating in Online Multiplayer Games, Diplomarbeit, Göteborg University, Göteborg. http://www.gamasutra.com/education/theses/20050610/zetterstrom_01.shtml [02. Dez 2006].

Pharmaforschung mit Open-Source-Methoden*

JANET HOPE

(CC-Lizenz siehe Seite 563)

In den letzten Jahren wurden innerhalb und außerhalb der Pharmaindustrie Befürchtungen laut, dass die pharmazeutische Forschung und Entwicklung von Arzneimitteln durch die hierarchischen Organisationsstrukturen vertikal integrierter Unternehmen immer starrer und unflexibler werden könnten. Ebenso bestehe die Gefahr, dass durch zu hohe Transaktionskosten, die bei der Koppelung verschiedener Forschungsergebnisse durch deren jeweilige proprietäre Lizenzen entstehen, die Forschung und Entwicklung ins Stocken geraten könnten. Eine Alternative zu den bisherigen Verfahrensweisen, sowohl im Umgang mit geistigen Eigentumsrechten als auch in der technologischen Entwicklung und den Vertriebsstrategien, wäre eine Arzneimittelentwicklung mit Open-Source-Methoden. Ein solcher Ansatz scheint zwar realisierbar, wäre jedoch aufgrund der unterschiedlichen technischen, rechtlichen und wirtschaftlichen Rahmenbedingungen in Arzneimittelentwicklung und Softwareentwicklung vergleichsweise schwierig umzusetzen. Noch viel schwieriger wird es allerdings sein, zu klären, welche Bedeutung Open-Source-Prinzipien im Bereich der Pharmaforschung – bzw. anderer technologischer Bereiche außerhalb der Softwareentwicklung – tatsächlich haben könnten.

Schlüsselwörter: Pharmaindustrie · Pharmaforschung · Transaktionskosten · Biotechnologie · Copyleft

1 Pharmaforschung am Scheideweg

Seit der Jahrtausendwende stehen die Erforschung und Entwicklung von Medikamenten an einem Scheideweg. So wurde im vorigen Jahrhundert die neu aufkommende

* Aus dem Englischen von Elke Eisenschmidt.

Pharmaindustrie zur Triebfeder innovativer interdisziplinärer Forschung auf den Gebieten der analytischen und synthetischen Chemie sowie der experimentellen Pharmakologie. Ende der neunziger Jahre hatte die Branche, obwohl sie weiterhin sehr hohe Gewinne erzielte, jedoch ihre Befähigung zu wahrhaft innovativer Forschung eingebüßt (Drews 1999, S. 229). Immer weniger der neu auf den Markt gebrachten Produkte stellten einen wesentlichen Fortschritt gegenüber der gebräuchlichen Medizin dar, während die Forschungs- und Entwicklungskapazitäten der Industrie kaum ausgelastet waren.[1]

Das soll nicht heißen, dass die Arzneimittelinnovation an einem absoluten Nullpunkt angekommen wäre. Das Innovationsdefizit konnte zum Teil von Universitäten und anderen gemeinnützigen Forschungseinrichtungen sowie den kleineren Unternehmen der neuen Biotechnologiebranche ausgeglichen werden. Zunehmend werden für Forschung und Entwicklung Kooperationsnetzwerke zwischen Einrichtungen unterschiedlichster Art gebildet, die es schaffen, alle nötigen Schritte von der Laborforschung bis zur klinischen Anwendung eines neuen Medikaments unter einem Dach zu vereinen (Powell 2001). Abstrahiert man von konkreten Projekten und Einrichtungen, ergibt sich folgendes Bild: Universitäten sorgen für die Grundlagenforschung, während Biotechnologieunternehmen sich weitestgehend mit der Wirkstoffforschung auseinandersetzen und Pharmaunternehmen Entwicklung, Vermarktung und Vertrieb übernehmen.

Diese neue Aufgabenverteilung ermöglicht zwar eine laufende Entwicklung sicherer und wirksamer neuer Medikamente, sie weist jedoch auch einige Schwachstellen auf. Insbesondere besteht ein Konflikt zwischen der technischen und institutionellen Realität und den bestehenden rechtlichen und wirtschaftlichen Rahmenbedingungen: Einerseits basieren pharmazeutische Forschung und Entwicklung von Arzneimitteln auf Kooperation und Wissenstransfer zwischen verschiedenen Forschungs- bzw. Entwicklungseinrichtungen, andererseits auf dem weltweiten Einsatz von Patenten, also rechtlich geschützten Monopolen, für den Vertrieb und Verkauf neuer Medikamente. So stehen im Zentrum eines jeden Kooperationsnetzwerks ein oder mehrere Patente, die es ermöglichen, andere von Herstellung, Nutzung, Verkauf oder Import des patentierten Produkts oder Verfahrens auszuschließen. Sobald neue Protagonisten (einschließlich gemeinnütziger Einrichtungen) die Bühne betreten, werden auch sie in das Spiel um die Patente hineingezogen.

Durch das immense Ausmaß der daraus resultierenden „Sperrräume" entstehen Transaktionskosten, die unweigerlich zu den sich ohnehin hochschraubenden Kosten für Forschung und Entwicklung beitragen (Heller und Eisenberg 1998). Die herrschenden Bedingungen in der heutigen Pharmaindustrie erlauben es Unternehmen, erfolgreiche Entwicklungen aus dem Innovationsprozess „abzuschöpfen", ohne dass sie diesem selbst wieder etwas hinzuführen müssen. So werden Forschung und Entwicklung von Unternehmen nur unterstützt, wenn diese die Chance sehen, mit den

1 Für einen generellen Überblick siehe Angell (2004) und Goozner (2004).

eventuell resultierenden Produkten einen Massenmarkt zu erreichen. Folglich wird der Bedarf all jener Patienten, deren Erkrankungen in den Industrienationen nicht weit verbreitet sind, kaum berücksichtigt, während die Forschung vielversprechende Wege gar nicht erst einschlägt.

2 Open Source als Alternative?

Unter den beschriebenen Umständen drängt sich die folgende Frage geradezu auf: Könnte Pharmaforschung mit Open-Source-Methoden in Technologieentwicklung, Lizenzierung und gewerblicher Nutzung in Zukunft zumindest eine Teillösung des Problems effizienter Organisation der Medikamentenentwicklung darstellen?

Erstens lassen sich unter günstigen Voraussetzungen durch Open-Source-Methoden – mitunter auch *commons-based peer production* und *bazaar governance* genannt – im Vergleich zu Firmen-, Markt- und Netzwerkmodellen bei geringerem Einsatz von Ressourcen innovativere Ergebnisse erzielen (Benkler 2002; Demil und Lecocq 2006). Somit erscheinen sie wirtschaftlicher als vertikal integrierte Forschung und Entwicklung innerhalb eines Pharmaunternehmens oder Kooperation verschiedener Firmen auf Basis herkömmlicher Eigentumsrechte.

Zweitens sind Open-Source-Lizenzen ein Mittel zum Zweck im iterativen Prozess der offenen Technologieentwicklung. Ihre Gestaltung basiert daher grundsätzlich auf dem Prinzip der Wechselseitigkeit, auch Copyleft genannt.[2] Zudem bemüht man sich beim Formulieren von Open-Source-Lizenzen auch um rechtliche und sprachliche Eindeutigkeit. Sowohl die Reziprozität als auch die Rechtssicherheit, die mit den Lizenzen einhergehen, senken zumindest potenziell die Transaktionskosten, welche durch den Erwerb von Zugangs- und Nutzungsrechten an Grundlagentechnologien in der Pharmaindustrie entstehen.

Schließlich wäre eine Vermarktung neuer Technologien auf Open-Source-Basis grundlegend gegen Monopolisierung gerichtet. Selbst wenn nur wenige Akteure der Biotechnologie- und Pharmabranche auf diese Strategie zurückgreifen würden, könnte das auf dem Markt für Arzneimittel und Werkzeuge der Arzneimittelentwicklung langfristig den Wettbewerb steigern, was den Verbrauchern bezüglich Preis und Verfügbarkeit zugute kommen würde.[3]

Alle diese Überlegungen haben in den letzten Jahren das Interesse am möglichen Einsatz von Open-Source-Praktiken in verschiedenen Bereichen der Arzneimittelforschung stark anwachsen lassen. Das einfachste Beispiel hierfür ist die Verwendung von Open-Source-Software in der Bioinformatik, wie *BioPython*, *BioPerl* oder das weit verbreitete *Basic Local Alignment Search Tool (BLAST)* zum Abgleichen von

2 Siehe Rosen (2004, S. 106).

3 Für eine Analyse dieser Vorgänge im Software-Kontext siehe Casadesus-Masanell und Ghemawat (2006).

DNA- und Proteinsequenzen.[4] Derartige Tools sind für die Arzneimittelforschung und -entwicklung von großer und wachsender Bedeutung (siehe z. B. Jorgensen 2004).

Abgesehen von Software-Programmen gibt es bislang kaum praktische Anwendungen von Open-Source-Prinzipien in der Medikamentenentwicklung. Jedoch wurden bereits erste Schritte unternommen, um die nötigen Infrastrukturen für eine erfolgreiche Umsetzung auf diesem neuen Gebiet aufzubauen. Man könnte dagegen einwenden, dass eine verteilte Entwicklung nach dem Modell von Open-Source-Software aufgrund der wichtigen Rolle, die das Internet dabei spielt, auf nicht digitale Technologien nicht anwendbar sei. Dazu muss gesagt werden, dass die biologische und chemische Grundlagenforschung zur Arzneimittelentwicklung schon immer auf einer gemeinschaftlichen Produktion von offen zugänglichem Wissen beruhte – wie bei wissenschaftlicher Forschung üblich. Zudem hat das Internet die verteilte Wissensproduktion nicht nur im Software-, sondern auch in nahezu jedem anderen Bereich ermöglicht. Zweifellos stimmt es, dass der Austausch von unvollständig digitalisierten und digitalisierbaren Informationen über biologisches Material langsamer und teurer ist als der von Quelltexten. Ein sinnvoller Vergleich ist jedoch nicht zwischen Software- und pharmazeutischer Entwicklung zu ziehen, sondern zwischen den verschiedenen *Methoden* der pharmazeutischen Forschung und Entwicklung.

Tatsächlich wurden viele praktische Fragen, die eine Open-Source-Arzneimittelentwicklung aufwerfen würde, bereits erfolgreich durch *public-private partnerships* (*PPPs*) gelöst. Ein Beispiel hierfür ist das 1999 gegründete *Medicines for Malaria Venture (MMV)*, das sich mit Erforschung und Entwicklung neuer und preisgünstiger Mittel gegen Malaria befasst.[5] Derartige Unternehmungen agieren als virtuelle Pharmaunternehmen, die jedem die Möglichkeit bieten, etwas zu ihren Projekten beizutragen. Bei *MMV* werden beispielsweise alle eingehenden Vorschläge zunächst von einem Fachausschuss überprüft und die für finanzierungswürdig befundenen Vorschläge einem firmeninternen Projektleiter unterstellt. Mit der eigentlichen Forschung und Entwicklung werden eine Reihe von Einrichtungen wie Universitäten, Pharmakonzerne, Biotechnologieunternehmen und Forschungsinstitute beauftragt, die aus öffentlichen und privaten Geldspenden an *MMV* bezahlt werden. In jeder Phase der Entwicklung (Targetvalidierung, Identifizierung und Optimierung potenzieller Medikamentenkandidaten, vorklinische und klinische Entwicklung) prüft der Fachausschuss anhand der Daten, ob das Projekt weitergeführt werden soll (Munos 2006). Der Erfolg von *PPPs* zeigt, dass eine Entwicklung neuer Arzneimittel auf Open-Source-Basis prinzipiell möglich ist.

Die ersten expliziten Versuche, Open-Source-Verfahren in der biomedizinischen Forschung einzusetzen, fanden im Jahr 1999 statt. Im Wettrennen um die komplette Entschlüsselung des menschlichen Genoms kam man am britischen *Sanger Institute*

4 Genaueres über spezielle Open-Source-Software in der Bioinformatik findet sich auf der Webseite von der *Open Bioinformatics Foundation* unter http://open-bio.org/wiki/Main_Page.

5 Siehe http://www.mmv.org.

auf den Gedanken, bei der Veröffentlichung der genetischen Daten Open-Source–Lizenzen zu verwenden, so dass jeder, der seine eigenen Daten zu den öffentlichen Daten des Humangenomprojekts hinzufügt, zur Veröffentlichung seiner „verbesserten" Version verpflichtet wäre (Cukier 2003). Letztendlich entschied man sich gegen eine solche Copyleft-ähnliche Lizenz mit der Begründung, jegliche Aussage über geistige Eigentumsrechte könne andere Teilnehmer und Unterstützer des öffentlichen Entschlüsselungsprojekts irritieren (Sulston und Ferry 2002, S. 212).

Dennoch wurde durch diesen Vorschlag der Weg für spätere Initiativen geebnet. Anfang 2005 liefen bereits mindestens zwei medienwirksame Versuche, einzelne Open-Source-Prinzipien auf die biomedizinische Forschung zu übertragen. So übernahm das internationale Haplotype-Mapping-Projekt (*HapMap*) den Open-Source–Gedanken und führte für die Veröffentlichung seiner Daten eine Copyleft-ähnliche sogenannte Click-Wrap-Lizenz ein.[6] Die *Tropical Disease Initiative*, deren Ziel die Behandlung zu wenig erforschter Krankheiten wie Malaria ist, bedient sich des Open–Source-Prinzips zur dezentralisierten Identifizierung potenzieller Arzneimitteltargets und -kandidaten.[7]

Wahrscheinlich sind diese aus den Medien wohlbekannten Initiativen im Bereich der Arzneimittelentwicklung nur die Spitze des Eisbergs. So wurden Open-Source-Lizenzen bekanntlich auch bei den Verhandlungen für die geplante *Innovative Medicines Initiative* vorgeschlagen – ein EU-Partnerschaftsprojekt mit dem Ziel, Engpässe im Bereich der Arzneimittelentwicklung durch eine Zusammenarbeit zwischen öffentlichem Sektor sowie Biotechnologie- und Pharmaindustrie zu beseitigen.[8]

3 Die Umsetzung eines Open-Source-Ansatzes in der Arzneimittelentwicklung

Natürlich treten bei der Übertragung des Open-Source-Prinzips aus dem ursprünglichen technischen, juristischen und kommerziellen Kontext auf einen neuen Bereich zahlreiche Probleme auf. Pharmazeutische Forschung und Entwicklung von Medikamenten sind kostspieliger, risikoreicher und strenger reguliert als Softwareentwicklung und das Erkennen und Beheben von Fehlern in klinischen Versuchen ist von Haus aus teurer und zeitaufwändiger. Die wichtigste Aufgabe ist es jedoch, eine Vorstellung davon zu bekommen, was es für eine bestimmte Technologie grundsätzlich bedeutet, *Open Source* zu sein, und zwar unabhängig vom spezifischen und ursprünglichen Kontext.

6 Nähere Informationen zum jetzt abgeschlossenen HapMap-Projekt finden sich auf der Internetseite http://www.hapmap.org. Details zu den Bedingungen für einen Zugriff auf die Daten, inklusive der Copyleft-ähnlichen Click-Wrap-Lizenz, sind abzurufen unter http://www.hapmap.org/cgi-perl/registration.

7 Die Website der *Tropical Disease Initiative* erreicht man unter http://www.tropicaldisease.org. Siehe auch Maurer et al. (2004).

8 Siehe Roelants (2006).

Jede Übertragung ist abhängig von der Interpretation des Originals und der Wahl eines Ersatzes für alle Konzepte, die auf den neuen Kontext nicht direkt angewendet werden können. Wer sich im Bereich der Open-Source-Software auskennt, mag einige der Übertragungsversuche als legitime Variationen eines wiedererkennbaren Themas ansehen, andere wiederum als Verzerrungen oder gar als berechnende Aneignung der ursprünglichen Idee von Open Source.

In der Softwareentwicklung sorgen Organisationen wie die *Open Source Initiative* oder die *Free Software Foundation* für Einheitlichkeit, indem sie für Open-Source- und ähnliche Modelle freiwillige Lizenzstandards in Form von Richtlinien und/oder Definitionen veröffentlichen. Da es eine solche Standardisierung für den Bereich der biomedizinischen Forschung und Entwicklung nicht gibt, kann nicht ausgeschlossen werden, dass die Sprache der Open-Source-Software-Lizenzen zu Zwecken eingesetzt wird, die den eigentlichen Zielen der Bewegung entgegenstehen oder – ganz pragmatisch betrachtet – dass sie zu Verwirrung unter den Mitwirkenden und somit zu erhöhten statt verminderten Transaktionskosten führt.

Wahrscheinlich wird es Lizenzstandards ähnlich der Open-Source- oder Free-Software-Definition früher oder später auch für die Lizenzierung patentierter Biotechnologie geben. Bis dahin schlage ich als Ansatz für die aktuellen Auseinandersetzungen, die möglicherweise zur Entwicklung solcher Standards führen werden, drei allgemeine Prinzipien vor, die dem „Geist" von Open Source entsprechen, ohne an eine bestimmte Technologie oder Eigentumsordnung gebunden zu sein: glaubhafte Selbstverpflichtung, Wettbewerb und (optional) Copyleft.

3.1 Glaubhafte Selbstverpflichtung

Glaubhafte Selbstverpflichtung meint die freiwillige Beschränkung des eigenen Einflusses, um anderen die Mitarbeit zu ermöglichen. In Bezug auf Open-Source-Technologieentwicklung muss der Projektträger auf dieses Prinzip zurückgreifen, um potenziellen Nutzern zu garantieren, dass Zeit und Ressourcen, die sie in die Übernahme und Verbesserung der Technologie investieren, nicht von ihm oder Dritten zu opportunistischen Zwecken ausgenutzt werden.

Bei Open-Source-Software-Projekten wird dies teils durch informelle Mechanismen gewährleistet und teils dadurch, dass man die üblichen Urheberrechte dafür nutzt, jedem das Recht auf Verwendung, Vervielfältigung, Veränderung und Weitergabe der lizenzierten Technologie zu gewähren. Damit eine Technologie als *Open Source* gelten kann, muss sie also durch geistige oder andere Eigentumsrechte geschützt sein und zu Konditionen weitergegeben werden, welche zumindest rechtlich durchsetzbar erscheinen.

Die Anwendung des Prinzips der glaubhaften Selbstverpflichtung hat für die praktische Gestaltung von Open-Source-Unternehmungen in der Biomedizin zwei Konsequenzen:

Wahl zwischen Patent und Allgemeingut Erstens ist eine unter Open-Source-Lizenz zur Verfügung gestellte Technologie per Definition nicht Gemeineigentum, sondern vielmehr Eigentum des Lizenzgebers.Soll also eine Technologie der Allgemeinheit zu niedrigen oder keinen Kosten zur Verfügung gestellt werden, so muss der Entwickler entscheiden, ob sie generell ungeschützt bleiben soll (was letztlich im o. g. Falle der Humangenomsequenz entschieden wurde) oder als Voraussetzung für eine Open-Source-Lizenzierung rechtlich geschützt werden muss. Diese zwei Optionen schließen sich immer und überall gegenseitig aus und haben beide sowohl Vor- als auch Nachteile. Entscheidend ist hier, dass die entsprechende Balance in der Biomedizin anders ausfallen wird als im Softwarebereich. Ein Patentschutz besteht im Gegensatz zum Urheberrecht weder automatisch noch kostenlos und ist daher bei neuen Biotechnologien ein erheblicher Schritt. Hier ist eine kostengünstigere Alternative, die Technologie derart zu veröffentlichen, dass sie nicht mehr Bestandteil von Patenten werden kann und somit der Allgemeinheit vollständig zur Verfügung steht. Die Gemeinfreiheit wiederum kann nicht ausreichend davor schützen, dass andere die neuen Technologien weiterentwickeln, ohne dann selbst ihre Ergebnisse wieder frei zugänglich zu machen. So drängten nach der direkten Veröffentlichung der Humangenomdaten durchaus Forscher auf die Patentämter, die nur in geringem Maße zur verbesserten Nutzbarkeit der Daten beigetragen hatten. Wenn die Zugänglichkeit nachgelagerter Technologieentwicklungen wichtig ist, ist daher selbst für patentfähige Erfindungen ein Open-Source-Ansatz erstrebenswert. Die Open-Source-Medikamentenentwicklung der *Tropical Disease Initiative* ist also streng genommen gar nicht Open Source, da sie zwar eine dezentralisierte Technologieentwicklung fördern und erleichtern will, jedoch keine konkreten Vorgaben zum Umgang mit geistigen Eigentumsrechten macht. Sie könnte es allerdings werden, wenn die Träger einzelner Forschungsprojekte eigene Open–Source-Lizenzen entwickeln.

Juristische Unanfechtbarkeit Zum Zweiten muss eine Lizenz, um nach dem Prinzip der glaubhaften Selbstverpflichtung zu funktionieren, natürlich von beiden Parteien als bindend angesehen werden. Welche Anforderungen eine Lizenz erfüllen muss, um dies zu gewährleisten, ist von Fall zu Fall unterschiedlich. In der Vergangenheit war es u. U. möglich für Open-Source-Software-Lizenzen das Prinzip der Gegenseitigkeit zu realisieren, obwohl sie juristisch nicht unanfechtbar waren. Dies ist im Softwarebereich immer weniger möglich, in der rechtlich komplexen – um nicht zu sagen unerbittlichen – Welt der Pharmaindustrie aber völlig undenkbar. Eine wirksame Open-Source-Lizenz für den Pharmabereich sollte denselben Anforderungen genügen wie andere Lizenzen in diesem Gebiet und dabei für Nicht-Juristen leicht verständlich bleiben. Als Mindestmaß sollte sie potenziellen Zuarbeitern einen angemessenen Schutz vor Missbrauch durch den Lizenzgeber bieten.

Die größte Hürde für die Entwicklung von Open-Source-Lizenzen für die Biotechnologie (abgesehen von Software-Lizenzen, die dort auch eingesetzt werden) ist bis heute die fehlende Erfahrung von potenziellen Lizenzgebern und -experten mit dem Open-Source-Konzept. Ist diese Hürde erst überwunden, so sind beim Verfassen von Open-Source-Lizenzen für Technologien, die einem anderen Urheberrechtsschutz als dem Copyright unterliegen, aber noch viele weitere Probleme zu lösen.

Zwischen den verschiedenen Eigentumsformen, die aus verschiedenen Eigentumsrechten resultieren, lässt sich konzeptuell keine eindeutige Übereinstimmung erkennen. Daher ist eine direkte Übertragung vertraglicher Bestimmungen von einem Kontext auf den anderen keine zuverlässige Vorgehensweise, um Lizenzgeber und -nehmer an vergleichbare Vertragsbestimmungen zu binden. Für die Praxis bedeutet dies, dass Open-Source-Lizenzen in der Biotechnologie von Grund auf neu erarbeitet werden müssen. Für das Erstellen konventioneller Lizenzen ist ein Verständnis der Biotechnologie, der damit verbundenen Eigentumsrechte, der geplanten Vertragskonditionen für Lizenzgeber und -nehmer und des allgemeinen rechtlichen Kontexts der Vereinbarung nötig. Man kann also schon hier von einem spezialisierten Vorgang sprechen.

Das Erstellen einer Open-Source-Lizenz für die Biotechnologie bedarf derselben Kenntnisse, doch darüber hinaus benötigt der Verfasser ein umfassendes Verständnis der Open-Source-Prinzipien sowie die Fähigkeit, Rechtskenntnisse kreativ einzusetzen und Bestimmungen so zu formulieren, dass sie nicht nur unanfechtbar, sondern auch für juristisch ungeschulte Biologen und Chemiker leicht verständlich sind. Da es für die Reduzierung von Transaktionskosten und die Durchsetzung des Prinzips der Wechselseitigkeit am sinnvollsten ist, wenn dieselben Konditionen immer wieder verwendet werden können, erfordert das Entwerfen einer soliden Open-Source-Lizenz für die Biotechnologie auch, von den jeweiligen Vorgängen zu abstrahieren und ein grundlegendes Verständnis für die allgemeinen Zusammenhänge zu haben. Besonders in den frühen Stadien der Lizenzentwicklung wird man nicht einfach, wie Juristen oder Lizenzverfasser es beim Formulieren neuer Verträge oft tun, auf Vorgänger zurückgreifen können. Diese Hindernisse sind keineswegs unüberwindbar, doch sie verdeutlichen den Bedarf an beträchtlichen Vorausinvestitionen – nicht nur in Form von Geld, sondern auch Zeit, Aufmerksamkeit und Kreativität von Spezialisten.

3.2 Wettbewerb

Das zweite allgemeine Prinzip der Open-Source-Lizenzierung ist der Wettbewerb. Hier geht es um gleiche Wettbewerbsbedingungen für den Lizenzgeber und andere Nutzer oder Zuarbeiter von Open-Source-Technologien was das Recht zur Verwendung und gewerblichen Nutzung der Technologie selbst sowie jeglicher nachgelagerter Neuerungen betrifft. Laut Lawrence Rosens Untersuchung darf eine Open-Source-Lizenz die Nutzung einer Technologie weder auf einen bestimmten Anwendungsbereich noch auf einen bestimmten Ort beschränken. In proprietären Lizenzen sind solche Einschränkungen üblich, um die Konkurrenzlosigkeit des Urhebers oder anderer Li-

zenznehmer in einem bestimmten Marktsegment zu gewährleisten. Weiter darf sie keine Melde- bzw. Rechenschaftspflicht an den Lizenzgeber oder die Bekanntgabe der internen Verwendung der lizenzierten Technologie fordern und keine einschränkenden Angaben zu Produktvertrieb, zu Identität oder Niederlassungsort der Kunden oder zur geforderten Lizenzgebühr (die überall zwischen Null und dem höchstmöglichen Marktpreis liegen kann) machen. Dasselbe gilt für Verbesserungen oder andere nachgelagerte Verwendungen, mit der Einschränkung, dass der Lizenznehmer diese unter einer Copyleft-Lizenz zu den selben Bedingungen an Dritte weitergeben muss, zu denen er sie selbst erhalten hat (Rosen 2004, S. 1–11).

Das Prinzip des freien Wettbewerbs zwischen Lizenzgeber und -nehmer ist ein zentraler Aspekt für die Schaffung von gemeinschaftlichen Entwicklungsprojekten und wird im Open-Source-Softwarebereich zum Teil als maßgebliches Charakteristikum betrachtet (Bedell 2004). In der Pharmaindustrie war freier Wettbewerb lange Zeit verpönt. Pharmakonzerne stützen sich gezielt auf Patentrechte, um in den Jahren nach den langwierigen und teuren vorklinischen und klinischen Studien und der Zulassung neuer Medikamente allen Wettbewerb auszuschalten. Der Wert dieser Ausschließlichkeitsrechte für den Patentinhaber lässt sich daran erkennen, dass nach Auslaufen eines Patents die Preise für ein Medikament durch die Konkurrenz von Generikaproduzenten um ungefähr zwei Drittel fallen (Angell 2004, S. 9). Das Festhalten an monopolistischen Praktiken hat neben den finanziellen jedoch noch tiefgreifendere Gründe. Es ist in der Tat Bestandteil einer Tradition, die sehr weit zurückreicht. Die wichtigsten Vorläufer der heutigen Pharmaindustrie waren die großen europäischen Apotheken des 18. Jahrhunderts, die aus lukrativen Monopolen über die Arzneimittelbelieferung von großen Handelsgesellschaften und des Militärs hervorgingen (Dukes 2006, S. 5 ff.) sowie die Chemiegesellschaften, die im 19. Jahrhundert aus Steinkohleteer-Derivaten neue aromatische Moleküle erzeugten und die größten und berüchtigsten Kartelle ihrer Zeit bildeten (Drahos und Braithwaite 2002, S. 151).

Auch Biotechnologieunternehmen sind, wenn auch aus etwas anderen Gründen, stark von Ausschließlichkeitsrechten durch Patente abhängig (Grubb 2004, S. 409–412). Die meisten der Firmen erzielen durch Verkaufs- und Lizenzerlöse nicht genügend Gewinne und sind bei der Mittelbeschaffung für laufende Forschungs- und Entwicklungsprojekte vom Kapitalmarkt abhängig. Die meisten Investoren (aber nicht alle, wie die erfolgreiche Beschaffung von Risikokapital durch einige Open–Source-Software-Unternehmen Ende der 90er Jahre zeigte) bestehen auf Patente, die eine qualitativ hochwertige Forschung, eine starke Marktposition und veräußerbares Vermögen im Falle eines Scheiterns gewährleisten sollen. Für Unternehmen, deren Exit-Strategie die Übernahme durch ein größeres Pharmaunternehmen oder deren Haupteinnahmequelle die Bereitstellung von lizenzierten Technologien ist, ist es ebenso wichtig, exklusive Nutzungsrechte vergeben zu können.

Impliziert diese Abhängigkeit von exklusiven Eigentumsrechten, dass in der Biotechnologie- und Pharmabranche ein Wettbewerb nach Open-Source-Prinzipen nicht

möglich ist? Nicht unbedingt. Für große Pharmakonzerne ist Anbieterbindung ein ernsthaftes Problem, wie es ein Biotechnologieexperte zum Ausdruck bringt:

> „Große Unternehmen mögen zwar nicht gerne teilen, aber noch weniger wollen sie in ihren Kerngeschäften von einem einzelnen Anbieter abhängig sein."[9]

Für Pharmaunternehmen als Leistungsnutzer wäre Wettbewerb unter den Leistungsanbietern also ein sehr großer Vorteil. Zweifellos war dies einer der Gründe für große Pharmakonzerne, die öffentliche Genomforschung auch unter der Bedingung einer Veröffentlichung ihrer Ergebnisse zu unterstützen. Ein weiterer Grund waren sicher – ähnlich wie im Bereich der Softwareentwicklung – die Vorteile vorwettbewerblicher Zusammenarbeit, wie Kosten- und Risikoteilung bei der Entwicklung von Basistechnologien, die nicht der Differenzierung im Wettbewerb dienen. Sicher kann man von einem Pharmakonzern nicht erwarten, seinen neuesten Kassenschlager unter eine Open-Source-Lizenz zu stellen, es ist aber sehr wohl denkbar, dass Pharmakonzerne in ihrer Rolle als Kunden die Open-Source-Technologieentwicklung unterstützten. Ebenso würde kein Biotechnologieunternehmen seine technologischen Weiterentwicklungen unter eine Open-Source-Lizenz stellen, solange es keine weitere verlässliche Einnahmequelle hat. Vorstellbar wäre jedoch, dass diese Firmen Weiterentwicklungen der Nutzer ihrer Technologien zusammentragen und in ihre eigenen Entwicklungen integrieren könnten.

Für Biotechnologie- und Pharmaunternehmen wäre ein Mitwirken an Open–Source-Entwicklungsprojekten auch in anderer Hinsicht wirtschaftlich sinnvoll: Der Austausch zwischen Grundlagenforschung und angewandter biomedizinischer Forschung ist über die Jahre immer enger geworden. Dass die pharmazeutische Forschung ins Stocken gerät, während Biotechnologiefirmen innovativ bleiben, liegt teilweise daran, dass die größeren etablierten Firmen weniger leicht Verbindungen zu Universitäten und anderen gemeinnützigen Forschungseinrichtungen aufbauen konnten (Drews 1999, S. 229). Da das Grundlagenwissen für die Arzneimittelentwicklung rapide zunimmt und verschiedenste Disziplinen umfasst, spielt die Lernfähigkeit eines Unternehmens, welche eine aktive Auseinandersetzung mit der neuesten Forschung voraussetzt, wirtschaftlich eine beträchtliche Rolle. Gleichzeitig ist aufgrund der starken Konkurrenz um Forschungspartner im hochgradig vernetzten Bereich der Biotechnologie und Pharmakologie ein guter Ruf als Kooperationspartner notwendig. Die Teilnahme an Open-Source-Projekten wäre somit nicht nur eine hervorragende Möglichkeit, seine Lern- und Kooperationsfähigkeit auszubauen, sondern auch eine Chance, anderen diese Qualitäten zu vermitteln.

Tatsächlich haben viele Unternehmen (vor allem in der Biotechnologie) diese Zusammenhänge erkannt. Deshalb, sowie auch um talentierte Mitarbeiter anzuziehen, erlauben sie ihren Wissenschaftlern, sich zehn bis zwanzig Prozent ihrer Arbeitszeit mit

9 Lee Bendeckgey, im Gespräch mit der Autorin, März 2003.

beliebigen Projekten ohne Bezug zu den internen Forschungszielen zu befassen oder bieten eine Art Forschungsurlaub an (Drews 1999, S. 229–232). Viele der angestellten Wissenschaftler könnten diese „freien Kapazitäten" gut für Open-Source-Projekte nutzen. Selbst wenn die so entwickelten Arzneimittel für die teilnehmenden Firmen wirtschaftlich uninteressant wären, so könnten sie in ihrer Gesamtheit doch von beachtlichem gesellschaftlichen Wert sein. Denn viele wissenschaftlich und therapeutisch wertvolle Projekte würden sonst, wie bereits erwähnt, an der äußerst hohen Rentabilitätsschwelle scheitern, die ein Unternehmen als Voraussetzung für seine Investitionen erwartet.

3.3 Copyleft

Das letzte Prinzip der Open-Source-Lizenzierung ist schließlich das Copyleft. Wie Leser dieses Jahrbuchs wissen, handelt es sich hierbei um ein wichtiges, wenn auch nicht notwendiges Merkmal von Open-Source-Softwarelizenzen. In den Biowissenschaften wurden die wechselseitigen Bedingungen von Open-Source-Lizenzen mit sog. Reach-through- und Grant-back-Bestimmungen in proprietären Lizenzen verglichen, wobei manche Beobachter befürchten, dass Open-Source-Lizenzen ähnliche Wettbewerbsverzerrungen wie diese nach sich ziehen könnten (Feldman 2004; Boettiger und Burk 2004). Solche Befürchtungen sind jedoch unbegründet, denn eine Analogie zwischen den genannten proprietären Bestimmungen und Copyleftregelungen hält einer genaueren Untersuchung nicht stand. Ein Open-Source-Lizenzgeber erhält zwar durch die Bestimmungen der Open-Source-Lizenz freien Zugang zu nachgelagerten Entwicklungen und kann von diesen direkt oder indirekt profitieren, er erhält aber kein exklusives Privileg. Er ist nur Teil der Öffentlichkeit, der mit der Open-Source–Lizenz das Recht auf freien Zugang und zur freien Nutzung der lizenzierten Neuerung verliehen wurde.

„Akademische" oder BSD-ähnliche Lizenzen haben für die Arzneimittelbranche sicher nützliche Eigenschaften, das Potenzial Copyleft-ähnlicher Lizenzen tritt allerdings viel deutlicher zu Tage. Man könnte einwenden, dass Pharma- oder Biotechnologieunternehmen diesem Ansatz aufgrund ihres vorherrschenden proprietären Denkens sehr reserviert gegenüberstehen würden. Selbst wenn sie durch das Mitwirken an oder die Unterstützung von Open-Source-Arzneimittelentwicklung (wie oben skizziert) profitieren könnten. Doch selbst dann würde die Open-Source-Arzneimittelentwicklung als Möglichkeit bestehen bleiben. Da die Privatwirtschaft auf die Arzneimittelforschung aus Steuermitteln an öffentlichen Einrichtungen angewiesen ist, könnte die dortige Einführung von Copyleft-ähnlichen Lizenzen mit der Zeit zu einer Veränderung der Wettbewerbsstrukturen führen. Kann die Privatwirtschaft keine eigenen neuen Arzneimittelkandidaten entwickeln, bliebe ihr dann nur übrig, Open-Source-lizenzierte Kandidaten weiterzuentwickeln. Ohne Monopolrechte müssten Pharmakonzerne miteinander sowie mit anderen Einrichtungen, die bereits vorklinische und klinische Tests von neuen Medikamenten durchführen, konkurrieren. Auch für den Staat könnten

sich (selbst hohe) Investitionen lohnen, um wirklichen Wettbewerb auf dem Arzneimittelmarkt zu etablieren, da dies niedrigere Ausgaben im Gesundheitswesen und die Entwicklung wirklich innovativer Medikamente zur Folge haben könnte.

4 Schlussfolgerung

Open-Source-ähnliche Lizenzen stellen eine mögliche Lösung für die Pharma- und Biotechnologieindustrie sowie deren Endkunden (Staat, private Krankenversicherung und Einzelpersonen) dar. Open-Source-Prinzipien sind zwar in der Entwicklungsmethodik, Lizenzierung und Vermarktungsstrategie für wichtige Bioinformatikprogramme in der Arzneimittelforschung und -entwicklung bereits deutlich erkennbar, ihre Anwendung auf andere biomedizinische Technologien, die zweifellos möglich ist, ist jedoch nicht so einfach. Ein wichtiger erster Schritt dabei ist, Open-Source-Prinzipien so zu formulieren, dass sie außerhalb des Softwarebereichs sinnvoll sind und gleichzeitig der immer noch im Entstehen begriffenen Bedeutung von Open Source treu bleiben.

Literatur

Angell, M. (2004), *The Truth About the Drug Companies: How They Deceive Us and What to Do About It*, Random House, New York.

Bedell, K. (2004), 'Open Letter to Sun from Eric Raymond', Blogeintrag auf ONLamp.com. http://www.oreillynet.com/onlamp/blog/2004/11/open_letter_to_sun_from_eric_r.html [09. Dez 2006].

Benkler, Y. (2002), 'Coase's Penguin, or, Linux and the Nature of the Firm', *Yale Law Journal* **112**(3), S. 369–446. http://www.benkler.org/CoasesPenguin.html [24. Nov 2006].

Boettiger, S. und Burk, D. L. (2004), 'Open Source Patenting', *Journal of International Biotechnology Law* **1**, S. 221–231. Verfügbar auf SSRN: http://ssrn.com/abstract=645182 [09. Dez 2006].

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

Cukier, K. N. (2003), 'Open Source Biotech: Can a Non-Proprietary Approach to Intellectual Property Work in the Life Sciences?', *The Acumen Journal of Life Sciences* **1**(3). http://cukier.com/writings/opensourcebiotech.html [09. Dez 2006].

Demil, B. und Lecocq, X. (2006), 'Neither Market nor Hierarchy nor Network: The Emergence of Bazaar Governance', *Organization Studies* **27**, S. 1447–1466.

Drahos, P. und Braithwaite, J. (2002), *Information Feudalism: Who Owns the Knowledge Economy?*, Earthscan, London.

Drews, J. (1999), *In Quest of Tomorrow's Medicines: An Eminent Scientist Talks About the Pharmaceutical Industry, Biotechnology and the Future of Drug Research*, Springer-Verlag, New York.

Dukes, G. (2006), *The Law and Ethics of the Pharmaceutical Industry*, Elsevier, Amsterdam.

Feldman, R. C. (2004), 'The Open Source Biotechnology Movement: Is it Patent Misuse?', *Minnesota Journal of Law, Science & Technology* **6**. Verfügbar auf SSRN: http://ssrn.com/abstract=545082 [09. Dez 2006].

Goozner, M. (2004), *The $800 Million Pill: The Truth Behind the Cost of New Drugs*, University of California Press, Berkeley.

Grubb, P. W. (2004), *Patents for Chemicals, Pharmaceuticals and Biotechnology*, Oxford University Press, Oxford.

Heller, M. A. und Eisenberg, R. S. (1998), 'Can Patents Deter Innovation? The Anticommons in Biomedical Research', *Science* **280**, S. 698–701.

Jorgensen, W. (2004), 'The Many Roles of Computation in Drug Discovery', *Science* **303**(5665), S. 1813–1818.

Maurer, S. M., Rai, A. und Sali, A. (2004), 'Finding Cures for Tropical Diseases: Is Open Source an Answer?', *Public Library of Science Medicine* **1**(3), S. e57–e59. DOI: 10.1371/journal.pmed.0010056, http://www.tropicaldisease.org/documents/MauRaiSal_PLOS2004.pdf [07. Jan 2006].

Munos, B. (2006), 'Can Open-Source R&D Reinvigorate Drug Research?', *Nature Reviews Drug Discovery* **5**, S. 723–729.

Powell, W. W. (2001), Networks of Learning in Biotechnology: Opportunities and Constraints associated with Relational Contracting in a Knowledge-Intensive Field, *in* R. C. Dreyfuss, D. Zimmerman und H. First (Hrsg.), 'Expanding the Boundaries of Intellectual Property: Innovation Policy for the Knowledge Society', Oxford University Press, Oxford, S. 251–266.

Roelants, I. (2006), The Innovative Medicines Initiative, Working paper, Catholic University of Leuven, Belgien. International Workshop on Gene Patents and Clearing Models: From Concepts to Cases.

Rosen, L. (2004), *Open Source Licensing: Software Freedom and Intellectual Property Law*, Prentice Hall, New Jersey. http://www.rosenlaw.com/oslbook.htm [09. Dez 2006].

Sulston, J. und Ferry, G. (2002), *The Common Thread*, Henry Joseph Press, Washington, DC.

Interaktive Wertschöpfung – Produktion nach Open-Source-Prinzipien

FRANK PILLER, RALF REICHWALD UND CHRISTOPHER IHL

(CC-Lizenz siehe Seite 563)

Die Produktionsprinzipien von Open-Source-Software sind nicht nur ein technisches Phänomen, sondern erweitern auch herrschende Theorien der Betriebswirtschaftslehre. Der Beitrag diskutiert, wie diese Prinzipien auch jenseits der Softwareproduktion (gegebenenfalls modifiziert) in Bereichen Anwendung finden können, die bislang durch eine unternehmensbezogene (hierarchische) Wertschöpfung gekennzeichnet sind. Wir nennen diese Übertragung interaktive Wertschöpfung: Ein Unternehmen vergibt in Form eines offenen Aufrufs eine Aufgabe, die bislang intern bearbeitet wurde, an ein undefiniertes (offenes), großes Netzwerk von Kunden und Nutzern. Die Bearbeitung dieser Aufgabe erfolgt dabei oft kollaborativ zwischen mehreren Nutzern, in anderen Fällen aber auch durch einen Akteur allein.[1]

Schlüsselwörter: Innovationsmanagement · commons-based peer production · Threadless · interaktive Wertschöpfung · Produktionsmanagement

1 Einleitung: Das Beispiel von Threadless

Open-Source-Software ist nicht nur ein technisches oder gesellschaftliches Phänomen, sondern das dahinter stehende Produktionsprinzip erweitert herrschende Theorien in der Managementforschung. Wir wollen in diesem Kapitel dieses Produktionsprinzip näher untersuchen und diskutieren anhand einiger Fallbeispiele, in welchen anderen Bereichen dieses Produktionsprinzip ebenfalls Anwendung finden kann. Im Mittelpunkt stehen dabei Strategien von Unternehmen, die ihre Kunden bzw. Nutzer nicht mehr nur als passive Empfänger und Konsumenten einer von Herstellern autonom

1 Der vorliegende Beitrag ist eine adaptierte Auskopplung aus dem Buch „Interaktive Wertschöpfung" (Reichwald und Piller 2006) der Autoren, das unter http://www.open-innovation.de auch zum Download unter einer CC-Lizenz zur Verfügung steht.

geleisteten Wertschöpfung sehen. Vielmehr treten Nutzer als Wertschöpfungspartner von Unternehmen oder anderen Nutzern auf, indem sie Produkte oder Dienstleistungen mitgestalten und teilweise sogar deren Entwicklung und Herstellung bestimmen oder übernehmen. Aus der von Unternehmen dominierten Wertschöpfung wird durch die aktive Rolle der Kunden eine interaktive Wertschöpfung (Reichwald und Piller 2006).[2]

Interaktive Wertschöpfung findet statt, wenn ein Unternehmen (oder eine andere Institution) eine Aufgabe, die bislang intern durch die Mitarbeiter bearbeitet wurde, an ein undefiniertes, großes Netzwerk von Kunden und Nutzern in Form eines offenen Aufrufs zur Mitwirkung vergibt. Offener Aufruf heißt dabei, dass die zu lösende Aufgabe offen verkündet wird und die externen Problemlöser durch Selbstselektion entscheiden, ob sie mitwirken oder nicht. Die Bearbeitung dieser Aufgabe erfolgt dabei oft kollaborativ zwischen mehreren Nutzern, in anderen Fällen aber auch durch einen Akteur allein. Die Aufgabe selbst kann sich dabei auf eine Innovation (Schaffung neuen Wissens), aber auch auf operative Aktivitäten (z. B. Mitwirkung beim Marketing oder bei der Konfiguration eines Produkts) beziehen. In jedem Fall aber wandelt sich die vom Unternehmen dominierte Wertschöpfung durch die aktive Rolle der Kunden und Nutzer zu einer Co-Kreation der resultierenden Leistung.

Ein konkretes Beispiel, wie wir interaktive Wertschöpfung verstehen, liefert das Unternehmen *Threadless*. Das im Jahr 2000 in Chicago gegründete Unternehmen verkauft mit großem Erfolg ein eigentlich einfaches Produkt: bedruckte T-Shirts. Die beiden Gründer und ihre knapp 20 Mitarbeiter erwirtschaften aber inzwischen pro Monat Gewinne in Höhe von fast einer halben Million Dollar – und das mit einer Handvoll von Mitarbeitern und ohne Entwicklungsrisiko (Ogawa und Piller 2006). Sie schaffen dies, da alle wesentlichen wertschöpfenden Aufgaben an die Kunden ausgelagert sind, die diesen mit großer Begeisterung nachkommen.[3] Einige Nutzer entwerfen neue Designs für T-Shirts und stellen sie auf die Website des Unternehmens. Derzeit werden etwa 800 neue Designs pro Woche hochgeladen (alle Angaben beziehen sich auf August 2006). Die meisten Nutzer aber haben weder Lust noch Zeit und vor allem nicht die Fähigkeit, selbst ein neues Design zu entwerfen. Sie sind aber dennoch nicht nur klassische Käufer, sondern übernehmen eine andere zentrale Aufgabe eines Modeunternehmens: das Produktmanagement.

Jede Woche bewerten mehr als 200 000 Nutzer die neuen Designs und machen Verbesserungsvorschläge zu den Entwürfen anderer. *Threadless* vertraut dabei seinen Nutzern völlig: Das Unternehmen produziert wöchentlich etwa vier neue Designs, und zwar die, die von der Mehrheit der Nutzer als besonders gelungen („I love it"

2 Hinweis: Unter einem Kunden verstehen wir den Abnehmer und vor allem Nutzer einer Leistung, unter einem Unternehmen den Anbieter und vor allem Hersteller der Leistung. Ein Kunde bzw. Nutzer kann dabei auch ein Unternehmen sein (im Business-to-Business-Geschäft). Bei der Leistung kann es sich sowohl um materielle Produkte als auch Dienstleistungen handeln.

3 Siehe http://www.chicagotribune.com/features/magazine/chi-0609100319sep10,1,2101701.story?ctrack=1&cset=true für eine ausführliche Darstellung.

in der Sprache von *Threadless*) bewertet wurden. Diese werden dann in hohen Auflagen gedruckt und für 15 Dollar verkauft, derzeit ca. 60 000 T-Shirts pro Monat. Die Kunden übernehmen für das Unternehmen dabei auch weitgehend das Marktrisiko, da sie sich zum Kauf eines Wunsch-T-Shirts (moralisch) verpflichten, bevor dieses in Produktion geht. Dieses *commitment* wird durch das Anklicken eines kleinen Buttons gegeben („I'd buy it"), der besagt, dass ein Nutzer ein Design nicht nur gelungen findet, sondern dieses auch kaufen würde. Die Kunden übernehmen weiterhin die Werbung für *Threadless*, stellen die Models und Fotografen für die Katalogfotos und werben neue Kunden.

Die Kunden fühlen sich dabei aber nicht etwa ausgenutzt, sondern zeigen im Gegenteil große Begeisterung für das Unternehmen, das ihnen diese Mitwirkung ermöglicht. Sie beschützen *Threadless* vor Nachahmern und übermitteln unzählige Ideen, wie das Unternehmen noch besser und produktiver werden kann. *Threadless* selbst fokussiert sich auf die Bereitstellung und Weiterentwicklung einer Interaktionsplattform, auf der die Interaktion mit und zwischen den Kunden abläuft. Das Unternehmen definiert zudem die Spielregeln, honoriert die Kunden-Designer, deren Entwürfe für eine Produktion ausgewählt wurden (der Urheber eines Gewinner-Designs erhält 2000 Dollar) und steuert den eigentlichen materiellen Leistungserstellungsprozess (Herstellung und Distribution).

2 Von Hierarchie und Markt zur interaktiven Wertschöpfung

Das Beispiel *Threadless* ist mehr als eine kreative Spielerei. Es ist eine völlig neue Art der Koordination arbeitsteiliger Wertschöpfung. Arbeitsteilung ist das Grundprinzip unseres Wirtschaftssystems. Sie hilft, Spezialisierungseffekte zu nutzen und komplexe Aufgaben effizient zu bewältigen. Gleichzeitig aber verursacht Arbeitsteilung auch Aufwand (Transaktionskosten). Deshalb suchen Ökonomen seit jeher nach effizienten Wegen, arbeitsteilige Prozesse effizient zu organisieren. Hierzu werden bislang zwei wesentliche Alternativen unterschieden: die hierarchische Koordination im Unternehmen (Erstellung einer Leistung im Unternehmen) oder die Nutzung des Marktmechanismus über Angebot und Nachfrage (Einkauf der Leistung am Markt). Eine Zwischenform bilden die verschiedenen Varianten von Unternehmensnetzwerken. Die heute immer noch dominierende Vorstellung, wie Unternehmen Werte schaffen, kann auf Prinzipien zurückgeführt werden, die vor 100 Jahren in der damals aufkommenden Industriegesellschaft entwickelt wurden. Vor allem Frederick Taylors Ansatz des *scientific management* legte mit seinem Fokus auf die Senkung von Produktionskosten die Basis für alle folgenden Debatten. Rationalprinzip, Güterknappheit und das Allokationsproblem kennzeichnen in seinem Modell die betriebswirtschaftliche Problemstellung von Organisation, Arbeitsteilung und Koordination der Wertschöpfung (Gutenberg 1951; Kosiol 1959). Fokus ist die effiziente Durchführung innerbetrieblicher Wertschöpfungsprozesse. Porters (1985) Modell einer Wertschöp-

fungskette präsentierte der Managementlehre einen integrierten Ansatz, wie man den Wertschöpfungsprozess von der Entwicklung über Produktion und Vertrieb bis hin zur Auslieferung von Gütern und Leistungen mit Hilfe des Produktionsfaktors Information organisieren und steuern kann. Anfang der 1990er Jahre wurde durch Hammer und Champy (1993) mit der Idee des *business process reengineering* in der Wirtschaft begeistert aufgenommener Ansatz vorgestellt, wie durch eine Fokussierung auf die interne Effizienz in einem Unternehmen Wert geschaffen werden kann, indem die Differenz zwischen der Zahlungsbereitschaft und den gesamten Herstellungskosten ausgeweitet wird.

Diese interne Sichtweise wurde später um das Bild eines grenzenlosen (oder gar virtuellen) Unternehmens erweitert, in dem ein eng verbundenes Netzwerk professioneller Akteure eine abgestimmte und friktionslose Wertschöpfungskette schafft, die viele Organisationen umfasst (Picot und Reichwald 1994; Picot et al. 2003). Die Zulieferer (und Zulieferer der Zulieferer) wurden in die Suche nach neuen effizienten Wertschöpfungsarrangements einbezogen (*supply chain management*). Mit dem Aufkommen des Internets und den daraus folgenden Potenzialen zur Senkung von Transaktionskosten wurden schließlich auch die Schnittstellen zu Abnehmern in die Effizienzbetrachtung einbezogen (*electronic commerce*). Entlang aller Stufen dieser Evolution steht dennoch stets die Annahme, dass das Streben nach interner (unternehmens- bzw. netzwerkinterner) Kosteneffizienz die Quelle betrieblicher Wertschöpfung ist.

Doch Kunden und Nutzer honorieren in der Regel nicht die interne operative Effizienz eines Anbieters bzw. Netzwerks. Sie mögen zwar günstige Preise als Resultat dieser Effizienz, doch hat sich gezeigt, dass das Streben nach immer weiterer operativer Effizienz innerhalb eines Netzwerks keine Quelle dauerhaft nachhaltiger Wettbewerbsvorteile ist (Porter 1996). Operative Effizienz ist eine notwendige, aber keine hinreichende Bedingung für dauerhafte Wettbewerbsvorteile. Vielmehr zeigt sich heute, dass vor allem die Gestaltung der Schnittstellen und der Aktivitäten an der Peripherie eines Unternehmens wesentliche Ansatzpunkte für die Schaffung von Wert bildet. Damit tritt auch ein Akteur in den Mittelpunkt der Betrachtung, der bislang in der Debatte um die Gestaltung der Wertschöpfung weitgehend ausgeblendet war: der Kunde bzw. Nutzer.

Wir sehen heute, dass Kunden das Ergebnis betrieblicher Wertschöpfung nicht nur konsumieren, sondern selbst einen wesentlichen Beitrag bei der Schaffung von Wert leisten (Ramirez 1999). Dies geschieht dabei nicht nur autonom in der Kundendomäne (ein Bereich, der in der Mikroökonomie schon lange im Zusammenhang mit Konsumentenproduktion untersucht wurde, siehe z. B. Becker 1965 und Lancaster 1966), sondern auch in einem interaktiven und kooperativen Prozess mit Herstellern und anderen Nutzern einer Leistung. Kunden und Nutzer tragen dazu bei, die Kenntnisse, Fähigkeiten und Ressourcen eines Herstellers zu erweitern (Gibbert et al. 2002). Interaktive Wertschöpfung heißt, die Kunden als strategischen Faktor in die

Aktivitäten eines Herstellers zu integrieren, die in einem erweiterten Wertschöpfungsnetzwerk Wert schaffen. Die Wahrnehmung dieses Wertes umfasst dabei weit mehr als die Erhöhung der Differenz zwischen Zahlungsbereitschaft und interner Effizienz. Ziel ist vor allem die gemeinsame Schaffung von Innovationen auf der Produkt- und Prozessebene.

Dazu wird eine Aufgabe, die bislang intern durch die Mitarbeiter eines Unternehmens (oder innerhalb eines geschlossenen Netzwerks klar definierter Partner) bearbeitet wurde, an eine offene, undefinierte und große Gruppe von Akteuren in der Peripherie des Unternehmens vergeben (Benkler 2002; Huff et al. 2006). Diese Vergabe erfolgt in Form eines offenen Aufrufs zur Mitwirkung. Das heißt, dass entweder das fokale Unternehmen oder ein Akteur innerhalb des Netzwerks ein Problem technischer oder organisatorischer Art formuliert („Suche ein tolles T-Shirt-Design", „Wie kann ich das T-Shirt waschen, ohne dass es einläuft?", „Wer tauscht sein Panda-T-Shirt gegen ein Starwars-T-Shirt?") und auf einer offen zugänglichen Plattform veröffentlicht. Alle potenziellen Akteure entscheiden dann selbst, ob sie mitwirken oder nicht (Selbstselektion). Die Lösung dieser Aufgabe erfolgt anschließend oft kollaborativ zwischen mehreren Nutzern (*peer production*), in anderen Fällen aber auch durch einen Akteur allein. Neu ist aber nicht nur die Aufgabenverteilung (Ausschreibung und Selbstselektion), sondern auch die Art und Weise, wie eine komplexere Aufgabe arbeitsteilig gelöst wird.

Denn entlang der Evolution der Organisation arbeitsteiliger Wertschöpfung ändert sich nicht nur die Sichtweise, welche Akteure am Wertschöpfungsprozess aktiv beteiligt sind, sondern auch die Vorstellung, wie das Organisationsproblem, d. h. die Koordination und Motivation der einzelnen Akteure, die die Gesamtaufgabe arbeitsteilig vollziehen, am besten gelöst werden kann: Taylors Modell setzt vor allem auf die hierarchische Koordination und Motivation durch finanzielle Anreize in einem geschlossenen Wertschöpfungssystem.

Die Netzwerkansätze erweitern diese Vorstellung um eine Kombination marktlicher und hierarchischer Koordinationsformen und betonen darüber hinaus auch eine Motivation durch nicht monetäre Anreize.

Die interaktive Wertschöpfung ergänzt diese beiden klassischen Koordinationsformen (Hierarchie und Markt) durch einen dritten Weg: die Selbstselektion und Selbstorganisation von Aufgaben durch (hoch) spezialisierte Akteure, deren Motivation vor allem die (eigene) Nutzung der kooperativ geschaffenen Leistungen ist, die jedoch durch eine Vielzahl weiterer sozialer, intrinsischer und extrinsischer Motive ergänzt werden kann.

Betrachten wir diese Prinzipien am Beispiel *Threadless*: In einem klassischen Modeunternehmen würde der Designchef dem am besten geeigneten Spezialisten mit der Aufgabe betrauen, ein neues T-Shirt zu einem bestimmten Motiv zu gestalten. Dazu würde er entweder unter seinen eigenen Mitarbeitern denjenigen aussuchen, den er für das Design am besten geeignet hält (bzw. vielleicht auch einfach nur den

Mitarbeiter, der gerade nicht ausgelastet zu sein scheint). Oder er würde in seinem Adressbuch suchen, um eine externe Designerin zu finden, die seines Erachtens das „beste" Design liefern kann. In beiden Fällen wird der Entwurf mit einem festen Preis entlohnt. Der Designchef wird einen festen Liefertermin nennen, damit der gesamte Produktionsprozess des Produkts nicht gefährdet wird, und die Einhaltung dieses Termins überwachen. Wird das Design geliefert, wird er es entweder akzeptieren, Verbesserungsvorschläge machen oder vielleicht auch ganz verwerfen („Das ist ja was ganz anderes, als was ich Ihnen im Briefing gesagt habe.") – er koordiniert die arbeitsteilige Wertschöpfung mittels hierarchischer oder marktlicher Kontrolle.

Bei *Threadless* dagegen gibt es keine Designchefs, die ihre Mitarbeiter anleiten, bestimmte Entwürfe zu gestalten. *Threadless* lädt jeden, der sich dazu berufen fühlt, ein, bei der Problemlösung mitzuwirken. Ob eine externe Designerin hierbei mitwirkt, wie viel Energie sie in die Lösung investiert und was sie dazu motiviert, bestimmt sie allein selbst. Der klassische hierarchische Koordinationsmechanismus wird durch Selbstmotivation, Selbstselektion und Selbstorganisation der Akteure ersetzt. Es gibt weder Hierarchien noch klassische Märkte. Jeder leistet den Beitrag, den sie oder er am besten leisten kann. Dies ist vor allem dann der Fall, wenn ein Teilnehmer bereits Wissen hat, das für eine Problemlösung weiterverwendet werden kann. Zum *Threadless*-Modell gehören aber nicht nur die Designer, sondern auch die tausenden „gewöhnlichen" Nutzer, welche die Bewertung der neuen Designs übernehmen, durch *affiliate marketing* die Produkte bewerben und natürlich letztendlich die Produkte kaufen (unsere Analyse von *Threadless* zeigt, dass so gut wie alle Kunden, die ein T-Shirt kaufen, auch vorher andere T-Shirts bewertet haben). Die Beiträge der gewöhnlichen Nutzer haben aus einer aggregierten Sicht einen genauso zentralen Anteil an einer effizienten Wertschöpfung des Unternehmens.

Das geht nicht nur bei T-Shirts. Ein klassisches Beispiel der *peer production* ist *Wikipedia*, wo die Teilnehmer selbst sowohl neue Beiträge in das Gesamtsystem integrieren als auch Ergänzungen und Verbesserungen bestehender Beiträge vornehmen. Dabei entscheiden alle Akteure selbst, warum und bei welchen Aufgaben sie mit welcher Intensität mitwirken. Bei *Wikipedia* ist auch die wichtige Aufgabe der Qualitätssicherung auf die Gesamtheit der Beitragenden ausgelagert. Basis der Qualitätssicherung ist dabei das Normensystem dieser Organisation. Doch auch hoch komplexe technische Produkte können nach diesen Prinzipien erstellt werden, wenn auch unter etwas anderen Rahmenbedingungen.

Ein gutes Beispiel dafür bietet *Innocentive*, ein amerikanischer Intermediär, der gegen Gebühr Probleme mit externen Problemlösern zusammenbringt. Der Name *Innocentive* ist ein Kunstwort, in dem Innovation (*innovation*) mit Anreiz (*incentive*) verschmolzen ist. Das Geschäftsprinzip von *Innocentive* ist einfach: Eine Firma sucht nach einer Lösung für ein Problem, das ihre Entwicklungsabteilung allein nicht lösen kann. Sie stellt diese Frage mit einer Beschreibung, Formeln oder Grafiken auf der Website von *Innocentive* dar und lobt ein Preisgeld aus, in der Regel zwi-

schen 10 000 und 100 000 Dollar. Das Preisgeld bekommt der Problemlöser, der die Aufgabe innerhalb eines vorgegebenen Zeitrahmens, z. B. zwei Monate, am besten löst. Mehr als 100 000 Tüftler haben sich bei *Innocentive* inzwischen registriert und lesen regelmäßig die neuen Aufgaben. Der Auftraggeber bleibt dabei anonym, um Firmengeheimnisse zu schützen. Im Gegenzug verlangt die Börse vom Fragesteller eine Gebühr. Seit ihrer Gründung im Jahr 2001 expandiert die Tüftlerbörse kräftig. Ursprünglich war sie eine Ausgründung des Pharmariesen *Eli Lilly*. Heute zählen zu den Kunden sogar konkurrierende Konzerne wie *BASF*, *Novartis*, *Nestlé* oder der Konsumgüterkonzern *Procter & Gamble*. Auch hier zeigt sich eine völlig neue Organisation der Wertschöpfung: Statt die Aufgabe an den „besten" bekannten internen oder externen Wissenschaftler zu vergeben, wird das Problem selbst offen ausgeschrieben. Ein jeder entscheidet selbst, ob er an der Lösungsfindung teilnimmt.

Eine Evaluierung der Erfolgsquoten von *Innocentive* zeigt, dass dieses offene Prinzip hoch effizient ist (Lakhani 2005): Mehr als die Hälfte aller Probleme werden schnell gelöst, obwohl in vielen Fällen die internen Abteilungen an der Lösung zunächst gescheitert waren. Oft gewinnen Problemlöser einen Wettbewerb, welche die Lösung vorher schon hatten. Dies zeigt ein weiteres wesentliches Prinzip interaktiver Wertschöpfung: die effiziente Wiederverwertung vorhandenen Wissens. Viele Gewinner hatten bereits in einer anderen Domäne eine Lösung für ein ähnliches Problem und können diese dann auf die neue Domäne übertragen.[4] Damit wird ein wesentliches Problem technischer Problemlösung überwunden: die „lokale" Suche nach Lösungen. Klassischerweise kann ein Unternehmen nur in den Bereichen nach Lösungen suchen, die es kennt bzw. nur solche Lösungen finden, die im Kompetenzbereich seiner Entwickler liegen. *Innocentive* überwindet – wie auch *Threadless* im Vergleich zum Designchef eines klassischen Modeunternehmens – diese Schranken „lokaler" Suche.

3 Voraussetzungen der commons-based peer production

Das hinter der interaktiven Wertschöpfung stehende Organisationsprinzip wurde vom Yale-Wissenschaftler Yochai Benkler (2002, 2006) als *commons-based peer production* bezeichnet: *peer production*, da eine Gruppe Gleichgesinnter (*peers*) gemeinschaftlich ein Gut produziert, *commons-based*, da das Ergebnis der Allgemeinheit zur Verfügung steht und auf offenem Wissen (*commons*) basiert. Unsere Idee der interaktiven Wertschöpfung baut auf der *commons-based peer production* auf, erweitert diese aber um einen Rahmen, in dem ein fokales Unternehmen diesen Prozess anstößt, moderiert oder unterstützt – genau wie wir es bei *Threadless* oder *Innocentive* gesehen haben.

4 Siehe Lakhani und von Hippel 2000 für eine Beschreibung des gleichen Mechanismus bei Open–Source-Software.

Damit diese interaktive Wertschöpfung funktioniert, müssen aber drei Bedingungen erfüllt werden: Erstens muss sich die Gesamtaufgabe in viele kleine Teilaufgaben spalten (Prinzip der *Granularität*) und einfach über eine Interaktionsplattform verteilen lassen. Denn nur so können die Hürde und der Aufwand für einzelne Nutzer gesenkt werden. Ziel ist, dass komplexe Aufgaben durch die verteilten Fähigkeiten vieler gelöst werden können, indem einzelne Nutzer vorhandenes Wissen optimal einbringen. *Wikipedia* zeigt dieses Prinzip ganz genau: Ein Nutzer könnte nie das gesamte Lexikon schreiben, die Mitwirkung tausender Teilnehmer mit jeweils kleinen Aufgaben (ein Stichwort schreiben, einen Fehler verbessern) bewältigt dagegen aus aggregierter Sicht eine gewaltige Aufgabe. Im Fall von *Threadless* geschieht dies durch die Zweiteilung der Mitwirkung in *Designer*, also Akteure, die bestimmte kreative Fähigkeiten haben, und *Bewerter*, d. h. (potenzielle) Kunden, die nur aufgrund ihres persönlichen Geschmacks entscheiden müssen. Bei *Innocentive* sorgen in der Regel genau abgrenzbare Teilprobleme (z. B. „Entwickeln Sie ein Molekül mit diesen Eigenschaften.“) für eine mögliche Zuordnung der Aufgabe zu den Fähigkeiten spezialisierter Wissensträger.

Zweitens müssen ausreichend viele motivierte Teilnehmer gewonnen werden können. Die Motivation der teilnehmenden Kunden und Nutzer ist einer der zentralen Aspekte der interaktiven Wertschöpfung. Denn die Ökonomie geht von rational Handelnden aus, die nur dann etwas beitragen, wenn sie dafür auch einen Gegenwert bekommen. Materielle Anreize fehlen bei den genannten Beispielen aber teilweise völlig. Bei *Threadless* stellen mehr als 800 Designer jede Woche neue Designs auf die Seite, gewinnen können aber nur drei bis vier. Was motiviert die Designer, hier mitzuwirken? Zum einen natürlich durchaus der Anreiz auf das Preisgeld, das mit 2000 Dollar ca. 4-mal so hoch ist wie das übliche Honorar für ein T-Shirt-Design bei einem klassischen Auftrag. Zum anderen aber vor allem die Möglichkeit, sich in einem Wettbewerb mit andern zu messen und dabei auch die eigenen Designfähigkeiten zu zeigen. Denn für einen (unbekannten) Grafikdesigner ist es sehr schwer, im Markt bekannt zu werden. *Threadless* bietet ihnen hier eine Plattform, ihre Arbeiten zu zeigen und potenzielle Auftraggeber auf ihre Fähigkeiten aufmerksam zu machen. Außerdem schätzen die Designer das Feedback, das sie von den Nutzern auf ihre Designs bekommen (zu jedem T-Shirt gibt es nicht nur eine Punktbewertung, sondern auch ein Forum, wo die Bewerter bis zu 90 Kommentare hinterlassen, von reinen Gefälligkeitsbemerkungen bis zu detaillierten Ideen, wie sich ein Design noch verbessern ließe).

Bei *Innocentive* scheint die Motivation auf den ersten Blick durch das hohe Preisgeld begründet zu sein. Jedoch zeigt Lakhani (2005) in einer Dissertation am MIT, dass die Mitwirkenden neben der Aussicht auf einen Preis vor allem auch durch den Wettbewerbscharakter motiviert sind: Sie wollen nicht nur mit ihrem Wissen Geld verdienen, sondern es reizen sie ebenso die Aufgabe des Tüftelns und die Möglichkeit, sich mit anderen Wissenschaftlern aus aller Welt zu messen.

Die inzwischen recht umfangreichen Arbeiten zur Motivation von Open-Source–Programmierern zeigen, dass bei aktiven Nutzern die Erwartung extrinsischen Nutzens (Honorar, Nutzung der geschaffenen Lösung) oft durch intrinsische Motive dominiert wird. Intrinsischer Nutzen bezieht sich auf die Ausführung einer Tätigkeit selbst. Eine Aktivität wird um ihrer selbst willen geschätzt und auch ohne unmittelbare Gegenleistung ausgeführt. So ist oft das Interaktionserlebnis selbst als solches positiv und nutzenstiftend, wenn es das Gefühl von Spaß, Kompetenz, Exploration und Kreativität vermittelt (Deci et al. 1999). Ebenso wird die Mitwirkung mit der Erfüllung sozialer Normen erklärt. Beispiele für eine solche Norm sind z. B. Reziprozität, Gemeinnützigkeit oder Fairness.

Die dritte Voraussetzung schafft den eigentlichen Vorteil einer interaktiven Wertschöpfung, ist aber in den Beispielen *Threadless* und *Innocentive* derzeit nur ansatzweise, wenn überhaupt, erfüllt: Offenheit und ein nicht proprietärer Schutz der geschaffenen Güter (*commons-based*). Nur wenn ohne aufwändige Lizenzierung auf vorhandenes Wissen zur Lösung neuer Probleme zurückgegriffen werden kann, kommt die Effizienz der neuen Art der Arbeitsteilung im Netz wirklich zum Tragen. Interaktive Wertschöpfung basiert so in ihrer Idealform auf der Offenlegung des geschaffenen Wissens der Beitragenden zur einfachen Nutzung, Kombination und Weiterentwicklung durch andere. Von Hippel (2005) sieht deshalb die bestehenden Patentsysteme als einen wesentlichen Faktor, der heute in etlichen Bereichen Innovation verhindert. Er verlangt Reformen, um die Wiederverwertung von vorhandenem Wissen zu erleichtern.

Threadless und *Innocentive* dagegen agieren noch im klassischen System intellektueller Schutzrechte: Ein Problemlöser bei *Innocentive* muss nachweisen, dass er das geistige Eigentum an einer Lösung hat (am besten durch ein vorhandenes Patent), um dieses dann gegen Zahlung des Gewinns an den Urheber des Problems zu lizenzieren (eine wesentliche Aufgabe von *Innocentive* ist, genau diesen Transfer zu garantieren). Auch bei *Threadless* wird mit dem Gewinn die Übertragung des Copyrights für den T-Shirt-Druck abgegolten. Bei Open-Source-Software bestehen diese Schranken dagegen nicht. Hier kann zur Lösung eines Problems weitgehend auf bereits vorhandenes Wissen zurückgegriffen werden, um dieses in eine eigene Lösung zu integrieren. Dies ist auf lange Sicht viel effizienter, als jedes Mal die genauen Schutzrechte an einem Wissensbaustein klären zu müssen. Wir können aber an dieser Stelle nicht vertieft in diese Diskussion einsteigen und wollen nur die grundsätzlichen Mechanismen aufgreifen: Interaktive Wertschöpfung funktioniert auch innerhalb klassischer Schutzrechte, ist jedoch erst dann wirklich leistungsfähig, wenn neue offene Schutz- und Lizenzierungsmodelle gefunden werden.

4 Zwei Beispiele zur interaktiven Wertschöpfung: Statistik-Software von Stata und Kitesurfing

Ein Unternehmen, das systematisch auf die interaktive Co-Entwicklung seiner Produkte durch die Anwender setzt und dabei einen guten Weg zwischen Offenheit und Geschlossenheit der resultierenden Ergebnisse gefunden hat, ist *Stata Corp.*, ein Hersteller statistischer Software. Kunden bzw. Anwender von Stata-Software sind meist Wissenschaftler oder Entwickler, die das Programm für eine Vielzahl statistischer Tests anwenden. Die Software erlaubt dabei die einfache Programmierung neuer Tests, falls die vorhandenen Anwendungen in dem Programm eine bestimmte Aufgabe nicht ausreichend (elegant) lösen können. *Stata* hat deshalb seine Software in zwei Teile gespalten: in einen proprietären Teil, der die Grundfunktionen bereitstellt und durch das Unternehmen selbst weiterentwickelt wird (und durch eine klassische Software-Lizenz kostenpflichtig vertrieben wird), und in einen offenen Teil, zu dem die Gemeinschaft aller Nutzer wesentliche Beiträge in Form neuer statistischer Algorithmen und Tests leistet. *Stata* unterstützt diese Expertennutzer, indem es ihnen eine Entwicklungsumgebung und ein Online-Forum zur Verfügung stellt, wo die Nutzer ihre eigenen Tests austauschen, anderen Nutzern Fragen stellen und Entwicklungen anderer weiterentwickeln können (von Hippel 2005).

Da allerdings nicht alle Nutzer derart versiert sind oder ausreichende Programmierkenntnisse haben, hat *Stata* ein Prozedere entwickelt, um regelmäßig die „besten" bzw. populärsten Weiterentwicklungen aus der Nutzer-Community auszuwählen und in die nächste kommerzielle Release-Version zu integrieren. Diese Entscheidung wird allein im Hause *Stata* getroffen, dessen Softwareentwickler auch die ausgewählten Anwendungen der Nutzer verbessern und reibungslos mit der Standardsoftware integrieren. Diese zusätzliche Wertschöpfung durch das Unternehmen ist auch Anreiz für die Nutzer, ihre Eigenentwicklungen in der Regel ohne monetäre Gegenleistung *Stata* zur Verfügung zu stellen (denn das Motiv für die Eigenentwicklung war ja sowieso die Nutzung der eigenen Anwendung für die eigene wissenschaftliche Arbeit).

Wir wollen abschließend noch ein letztes Beispiel betrachten, bei dem die Nutzer eine klassisch organisierte Industrie durch eine vollständige *commons-based peer production* völlig gewandelt haben.[5] *Kitesurfing* ist eine der derzeit aufstrebenden Trendsportarten. Der Sport wurde von Surfern initiiert, die – getrieben von dem Wunsch nach immer höheren und weiteren Sprüngen – mit der Kombination eines Surfboards und eines Segels vom Drachenfliegen experimentierten. Aus diesen anfänglichen Versuchen entwickelte sich in den letzten Jahren eine beachtliche Nischenindustrie, die inzwischen viele Anhänger hat. Die Kitesurfing-Industrie ist ein Beispiel dafür, wie Kunden als Produktentwickler die Regeln industrieller Wertschöpfung ändern können. Sie tragen nicht nur entscheidend zur Entwicklung des Equipments bei, sondern übernehmen inzwischen auch viele andere Aufgaben, die früher in der Ver-

5 In Anlehnung an die Beschreibung dieses Beispiels in von Hippel (2005).

antwortung professioneller Hersteller gesehen wurden, allen voran die Koordination des Produktionsprozesses.

Diese Hersteller, oft gegründet von Sportlern, die ihr Hobby zum Beruf gemacht haben, bilden heute eine ca. 100-Millionen-Dollar-Industrie, die vor allem die *kites* (Drachensegel) entwickelt, produziert und vertreibt. Um ein neues Kite-Produkt erfolgreich umzusetzen, werden eine Vielzahl an Fähigkeiten benötigt: Kenntnisse über Materialien und deren Eigenschaften für die Segel, Kenntnisse über Aerodynamik und Physik für die Formen der Segel, Kenntnisse über Mechanik für die Seilsysteme etc. Die Hersteller sind bei der Entwicklung neuer Designs in der Regel auf die Kenntnisse beschränkt, die sie in ihrem eigenen Haus haben, meist kleine Entwicklungsabteilungen aus drei bis fünf Mitarbeitern. Das Ergebnis sind eher kontinuierliche Weiterentwicklungen und Verbesserungen bestehender Designs als radikal neue Entwicklungen.

Die Kunden dagegen haben ein viel größeres Potenzial zur Verfügung und keine Werksgrenzen zu beachten. Initiiert und koordiniert von einigen begeisterten Kite–Surfern existieren heute eine Reihe von Internet-Communities, in denen die Mitglieder neue Designs für Drachensegel gemeinsam entwickeln, veröffentlichen und kommentieren. Mit Hilfe einer Open-Source-Design-Software (eine Art CAD-System) können die Nutzer zum Beispiel auf zeroprestige.org[6] neue Designs für die *kites* entwerfen und zum Download bereitstellen. Anderen Nutzern dienen diese Designs als Ausgangslage für eine Weiterentwicklung. Vielleicht bekommen sie so aber auch die Idee für eine radikal neue Entwicklung. Unter den vielen hunderten teilnehmenden Nutzern arbeiten manche in ihrem Berufsleben mit neuen Materialien. Andere studieren vielleicht Physik oder sind gar als Strömungstechniker bei einem Autohersteller tätig. Oft kann diese Gruppe von Kundenentwicklern auf einen viel größeren Pool an Talenten und Fähigkeiten zurückgreifen, als dies einem Hersteller möglich ist. Das Ergebnis ist eine Vielzahl an neuen Entwicklungen, Tests, Modifikationen und schließlich neuer Designs für Drachensegel, die allen Mitgliedern der Community zur Verfügung stehen (unter einer Open-Source-Lizenz).

Kitesurfing ist ein besonders spannender Fall, da hier die Kunden als Anwender noch einen Schritt weiter gehen: Denn was nützt der innovativste neue Entwurf für einen neuen *kite*, wenn dieser nur als Datei existiert? Findige Kunden haben herausgefunden, dass an jedem größeren See ein Segelmacher existiert, der CAD-Dateien verarbeiten kann. Die Kunden können so ein Design ihrer Wahl herunterladen, diese *Datei* zum Segelmacher bringen und dort professionell in ein Produkt umsetzen lassen. Da dieser Prozess keinerlei Innovationsrisiko und Entwicklungskosten für den Hersteller beinhaltet, sind die derart hergestellten Drachen oft um mehr als die Hälfte billiger als die Produkte der professionellen Kite-Hersteller, und das bei oft überlegener Leistung. Die Koordinationsleistung des Produzierens wird dabei ebenfalls von den Anwendern übernommen. Setzt sich diese Entwicklung fort, ist leicht

6 http://www.zeroprestige.org

vorzustellen, dass die Kunden Teile dieser Industrie „übernehmen" werden bzw. die bisherigen Anbieter in eine reine „Produzentenrolle" drängen. Die Motivation der Nutzer ist dabei nicht Profitmaximierung oder die Marktführerschaft, sondern das Streben nach dem bestmöglichen Produkt zur Eigennutzung. Die Anwender, die sich an diesem Prozess beteiligen, haben verstanden, dass dieses Ziel am besten nicht durch einen geschlossenen, sondern durch einen offenen Innovationsprozess erreicht werden kann – in dem auch die Ergebnisse offen mit allen geteilt werden. Ihr eigenes Engagement ruft Reaktionen und Beiträge anderer hervor und schafft damit einen höheren Mehrwert für alle.

5 Ausblick: Grenzen und Chancen der interaktiven Wertschöpfung

Eine interaktive Wertschöpfung hat aber auch Grenzen, die wir hier abschließend kurz diskutieren wollen.[7] Aufgrund dieser Grenzen werden auch in Zukunft die klassischen Modelle zur Organisation arbeitsteiliger Wertschöpfung noch ihre Berechtigung behalten. Denn nicht alle Aufgaben, die in einer Wirtschaft zu erfüllen sind, lassen sich nach dem Modell der interaktiven Wertschöpfung lösen. Wesentliche Grenzen dieses Modells sind der *trade-off* zwischen *Granularität* und Aufgabenteilung, die nachhaltige Motivation der Teilnehmer und die Überwindung des Not-invented-here-Syndroms.

5.1 Trade-off zwischen Granularität der Aufgabenteilung und Transaktionskosten

Je besser sich eine Wertschöpfungsaufgabe für eine sehr feingliedrige Aufteilung eignet, d. h. je höher deren *Granularität* ist, desto leichter kann ein größerer Aufgabenumfang an ein Netzwerk aus Kunden und Nutzern externalisiert werden. Ebenso sind so leichter Spezialisierungseffekte im Netzwerk zu nutzen. Allerdings bedarf es der innerbetrieblichen Koordination und Integration der einzelnen Wertschöpfungsbeiträge, was bei einer feingliedrigen Aufgabenteilung hohe interne Kosten verursacht. Deshalb ist bei Aufgaben, die sich nicht einfach digital abbilden lassen, eine interaktive Wertschöpfung viel schwieriger als bei rein digitalen Gütern.

Doch ist in vielen Bereichen heute eine Trennung des Informationsanteils und des physischen Kerns recht einfach möglich: T-Shirts, ein klassisches materielles Produkt, wurden von *Threadless* zu einem Informationsgut gewandelt, das erst kurz vor der Produktion die digitale Gestalt verlässt. Gleiches gilt für die Drachensegel im Kitesurfing-Beispiel. Auf der anderen Seite jedoch ist das ambitionierte OScar-Projekt, das *Open Source car*, bislang weitgehend gescheitert. Problem scheint hier unter anderem die Schwierigkeit zu sein, einen komplexen technischen Vorgang wie eine Autoentwicklung allein im Netz zu koordinieren.

7 Siehe ausführlich Reichwald und Piller (2006).

5.2 Motivation der Teilnehmer

Eine weitere Grenze ist die Motivation der Teilnehmer. Bislang scheint die Begeisterung von Nutzern wie Unternehmen grenzenlos. Das Wissen der Kunden bzw. Nutzer wird als großer ungenutzter Wissenspool gesehen, den es nur abzuschöpfen gilt. Viele Unternehmen versuchen derzeit, teilweise recht unbeholfen, die Fähigkeiten ihrer Kunden und Anwender zu nutzen (einen Fall, wie man es nicht machen soll, beschreibt Piller (2006) am Beispiel des Unternehmens *Kraft*). Solange jedoch mit solchen Initiativen keine klaren Anreizstrukturen verbunden sind, werden die Nutzer nach einer Phase der Euphorie („Das Unternehmen hört mir ja endlich mal zu.“) schnell in eine Ernüchterung verfallen („Die saugen ja nur mein Wissen ab.“) und nicht mehr zur Mitwirkung bereit sein. Ein wesentlicher Punkt an dieser Stelle ist sicherlich die freie, nicht proprietäre Verfügbarkeit des resultierenden Wissens. Diese hat nicht nur eine produktive Wirkung, wie bereits oben diskutiert, sondern auch einen motivierenden Effekt (Fairness, vor allem aber sofortige Nutzungsmöglichkeit durch die Beitragenden). Eine Vielzahl der heute herrschenden Geschäftsmodelle bestehender Unternehmen verhindert jedoch diese Offenheit. Deshalb müssen Unternehmen sehr genau die Motive ihrer Kunden zur Mitwirkung kennen und diese durch entsprechende Anreize bedienen (vgl. Brockhoff 2005). Bei vielen Unternehmen scheinen dafür aber sowohl Wissen als auch Verständnis zu fehlen.

5.3 Not-invented-here-Problem

Ein wesentlicher Bestandteil eines funktionierenden Systems interaktiver Wertschöpfung sind aber auch geeignete Anreize im Unternehmen, um externes Wissen effektiv innerbetrieblich weiterverwenden zu können. Nur bei wenigen etablierten Unternehmen herrscht eine derartige Offenheit für den Input der Nutzer wie bei *Stata* oder *Threadless*. Für viele Manager ist die Vorstellung, dass Nutzer einen (besseren) Beitrag zur Weiterentwicklung der eigenen Produkte leisten können, sehr fremd (Huff et al. 2006). Oft sind es einige fortschrittlich denkende Abteilungen im Unternehmen, die eine Initiative zur Integration von Kundeninformation starten und Beiträge durch die Nutzer anregen. Dieser Input muss dann aber durch andere Abteilungen weiterverarbeitet und genutzt werden. Unter dem Begriff Not-invented-here-Syndrom (NIH) wird im Innovationsmanagement ein Problem diskutiert, das genau diesen Transfer betrifft. Katz und Allen (1982, S. 7) definieren dieses als:

> „[. . .] the tendency of a project group of stable composition to believe that it possesses a monopoly of knowledge in its field, which leads it to reject new ideas from outsiders to the detriment of its performance.“

Klassischerweise wurde das NIH-Phänomen unternehmensintern an den Schnittstellen zwischen verschiedenen Bereichen nachgewiesen (z. B. Widerstände der Entwicklungsingenieure, Input aus der Marketingabteilung zu berücksichtigen). Es ist

anzunehmen, dass Widerstände gegen externes Wissen oft noch größer sein werden als in Bezug auf den Input eigener Kollegen. Dies bedeutet im Falle einer interaktiven Wertschöpfung zwischen Kunden und einem Herstellerunternehmen, dass Wissen aus externen Quellen auf Widerstand bei wenigstens einem Teil der internen Nutzer dieses Wissens stoßen kann. Diese Grenze hat sich in vielen Unternehmen heute als ein wesentliches Hindernis erwiesen, eine interaktive Wertschöpfung dauerhaft zu verwirklichen.[8]

Interaktive Wertschöpfung kann deshalb nicht einfach als Outsourcing interner Aufgaben an die Peripherie gesehen werden, sondern verlangt vielmehr eine aktive Beteiligung durch den Anbieter, der hierfür bestimmte Ressourcen und Fähigkeiten besitzen muss. Diese Interaktionskompetenz konkretisiert sich in den Organisationsstrukturen, der Definition adäquater externer und innerbetrieblicher Anreizstrukturen und in Systemen und Werkzeugen der Information und Kommunikation, um den Interaktionsprozess effizient, effektiv und auch mit entsprechender *usability* ablaufen zu lassen. Bislang haben nach unserer Einschätzung nur wenige bestehende Unternehmen diese Interaktionskompetenz aufgebaut.

Deshalb ist interaktive Wertschöpfung bzw. die Übertragung der Prinzipien einer Open-Source-Softwareproduktion auf andere Bereiche auch kein dominierendes System, das alte Wertschöpfungssysteme über Nacht ablösen wird. Viele Unternehmen sind bei der Umsetzung der interaktiven Wertschöpfung erst ganz am Anfang. Man sollte aber nicht vergessen, dass auch die klassische Massenproduktion viele Jahrzehnte gebraucht hat, bis sie in modernen Produktionssystemen perfektioniert wurde. Genauso wird es auch noch etliche Zeit dauern, bis sich interaktive Wertschöpfung als breites Phänomen zeigt. Ein Faktor ist dabei jedoch anders: Bei klassischen Unternehmensformen, die dem Beharrungsvermögen des Management ausgesetzt waren, bestimmen heute die Kunden den Wandel und treiben diesen voran. Interaktive Wertschöpfung ist aber nicht universell anwendbar. Es handelt sich vielmehr um eine Ergänzung bewährter Ansätze und Instrumente des Innovations- und Produktionsmanagements. Die alten Prinzipien haben weiter Bestand, die neuen aber schaffen eine Grundlage für neue Wege zu Wettbewerbsvorteilen.

Literatur

Becker, G. S. (1965), 'A theory of the allocation of time', *Economic Journal* **75**, S. 493–517.

Benkler, Y. (2002), 'Coase's penguin, or, Linux and The nature of the firm', *Yale Law Journal* **112**(3), S. 369–446. http://www.benkler.org/CoasesPenguin.html [24. Nov 2006].

Benkler, Y. (2006), *The wealth of networks*, Yale University Press, New Haven.

Brockhoff, K. (2005), 'Konflikte bei der Einbeziehung von Kunden in die Produktentwicklung', *Zeitschrift für Betriebswirtschaft* **75**(9), S. 859–877.

8 Zu Gegenmaßnahmen siehe Reichwald und Piller (2006).

Deci, E. L., Koestner, R. und Ryan, R. M. (1999), 'Meta-analytic review of experiments: Examining the effects of extrinsic rewards on intrinsic motivation', *Psychological Bulletin* **125**(6), S. 627–668.

Gibbert, M., Leibold, M. und Probst, G. (2002), 'Five styles of customer knowledge management, and how smart companies use them to create value', *European Management Journal* **20**(5), S. 459–469.

Gutenberg, E. (1951), *Die Produktion*, Bd. 1 aus *Grundlagen der Betriebswirtschaftslehre*, Springer, Berlin, Heidelberg.

Hammer, M. und Champy, J. (1993), *Reengineering the corporation*, Harper Business, New York.

Huff, A. S., Piller, F., Moeslein, K. und Fredberg, T. (2006), 'Creating centripetal innovation capacity', Proceedings of the AOM 2006 Meeting.

Katz, R. und Allen, T. (1982), 'Investigating the Not Invented Here (NIH) syndrome: A look at the performance, tenure, and communication patterns of 50 R&D projects', *R&D Management* **12**, S. 7–19.

Kosiol, E. (1959), *Grundlagen und Methoden der Organisationsforschung*, Duncker & Humblot, Berlin.

Lakhani, K. R. (2005), The core and the periphery in self-organizing and distributed innovation systems, PhD thesis, MIT Sloan School of Management, Cambridge.

Lakhani, K. R. 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 [24. Nov 2006].

Lancaster, K. J. (1966), 'A new approach to consumer theory', *Journal of Political Economy* **74**(2), S. 132–157.

Ogawa, S. und Piller, F. (2006), 'Reducing the risk of new product development', *MIT Sloan Management Review* **47**(2), S. 65–71.

Picot, A. und Reichwald, R. (1994), 'Auflösung der Unternehmung? Vom Einfluss der IuK-Technik auf Organisationsstrukturen und Kooperationsformen', *Zeitschrift für Betriebswirtschaft* **64**(5), S. 547–570.

Picot, A., Reichwald, R. und Wigand, R. (2003), *Die grenzenlose Unternehmung*, 5. Aufl., Gabler, Wiesbaden.

Piller, F. (2006), 'Kraft Foods Crowdsources its Innovation Process to its Customer', Beitrag im Blog „Mass Customization & Open Innovation News". http://mass-customization.blogs.com/ [03. Jun 2006].

Porter, M. (1985), *Competitive advantage: creating and sustaining superior performance*, The Free Press, New York.

Porter, M. (1996), 'What is strategy?', *Harvard Business Review* **74**(6), S. 61–78.

Ramirez, R. (1999), 'Value co-production: intellectual origins and Implications for practice and research', *Strategic Management Journal* **20**(1), S. 49–65.

Reichwald, R. und Piller, F. (2006), *Interaktive Wertschöpfung: Open Innovation, Individualisierung und neue Formen der Arbeitsteilung*, Gabler, Wiesbaden. http://www.open-innovation.de/ [24 Nov 2006].

von Hippel, E. (2005), *Democratizing Innovation*, MIT-Press, Cambridge. http://web.mit.edu/evhippel/www/books.htm [24. Nov 2006].

Piazze telematiche, Video Bridges, Open Coops – der mühsame Weg zu den Globalen Dörfern

FRANZ NAHRADA

(CC-Lizenz siehe Seite 563)

Kann das Prinzip „Open Source“ eine neue auf kooperativen Prinzipien basierende Produktionsweise über die Wissensproduktion hinaus tragen? Ja, so der Autor, wobei er auf die Biologie der Erbinformation hinweist, die ein sich ständig selbst modifizierendes „Biolinux“ darstellt und Träger der automatisierten Fülle des Lebendigen ist. Globale Dörfer kündigen das Ende des Kartesischen Zeitalters an, in dem ein Zentrum die Peripherie gestaltet. Im Zeitalter der Netze hat die Welt keinen privilegierten Mittelpunkt mehr.

Schlüsselwörter: Globale Dörfer · Open-Source-Produktion · Stoffkreisläufe

1 Einleitung

Während „Open Source“ im Bereich der Software fest etabliert ist und im Bereich von Wissen und Kultur als explizites Bekenntnis zu Wissen als Gemeingut zumindest eine deutlich in der Öffentlichkeit diskutierte Option und Strategie ist, fehlt es immer noch an überzeugenden Konzepten für Open Source im Bereich der materiellen Produktion. Nicht nur das, schon die Frage nach solchen Konzepten scheint vollkommen verfehlt zu sein. Selbst jene, die die Nichtmarktförmigkeit in der Produktionsweise freier Software betonen, tun sich schwer mit der Vorstellung einer Übertragung auf Bereiche der materiellen Produktion.[1] Die Produktion von materiellen Gütern gehorche nun einmal völlig anderen Gesetzmäßigkeiten als die Produktion von Wissen oder das Kopieren von Symbolketten. Sie basiere auf Knappheit von Ressourcen und hohen

1 Stellvertretend für viele: „Why do we rely almost exclusively on markets and commercial firms to produce cars, steel, and wheat, but much less so for the most critical information our advanced societies depend on? Is this a historical contingency, or is there something about information as an object of production that makes nonmarket production attractive?“ (Benkler 2006). Das Oekonux-Projekt versucht sich seit Jahren an der Zurückweisung dieser Opposition zwischen materiell und immateriell, hat bis heute freilich keine eindeutigen Ergebnisse geliefert (vgl. hierzu Merten und Meretz 2005).

Transaktionskosten, die nicht wie bei Information gegen Null abgesenkt werden könnten.

Eine Aufzählung der schon vorhandenen Beispiele für freie Hardware im Sinne von Bauplänen technisch vollständig dokumentierter und jederzeit rechtlich abgesichert reproduzierbarer Hardware, die die Existenz von Beispielen der Übertragung der Prinzipien von Open Source auf Hardware belegen, ist zwar möglich, aber sie lässt die Frage offen, ob es sich dabei um Ausnahmen oder Vorboten einer künftigen Regel handelt.[2]

In diesem Aufsatz versucht der Autor einen anderen Zugang zu wählen: ein in sich konsistentes und zugleich zugegebenermaßen noch visionäres Modell einer auf Open–Source-Prinzipien basierenden Produktionsweise vorzustellen, in dessen Kontext sich das freie Wissen quasi naturwüchsig als Leitprinzip ergibt.

2 Biolinux oder: Die Open Source des Lebens

Unsere gesamte biologische Welt beruht auf der ständigen Freisetzung und Reproduktion von genetischem Code. Noch bevor irgendein Mensch einen Gegenstand hergestellt hat, existierte für Jahrmilliarden bereits eine Reproduktionstechnologie von ungeheurer Komplexität, die sich im freien Spiel der Evolution beständig verfeinert und weiterentwickelt hat: die Vervielfachung der lebendigen Organismen, von der einfachen Zellteilung bis hin zur sexuellen Fortpflanzung. Manche Naturprozesse tendieren zu unendlicher Wucherung, die nur in der Erschöpfung von natürlichen Ressourcen ihre Grenzen findet, andere haben selbsteingebaute Grenzen. Die Welt der Pflanzen und Tiere weist vom stofflichen Gesichtspunkt eine ungeheure Diversität und Materialreichtum auf – die Natur ist darin noch immer den besten Baumeistern und Chemikern zumindest ebenbürtig.

Der Mensch ist immer noch am Anfang seiner Bemühungen, das Funktionieren dieser Reproduktionstechnologie zu verstehen, und vielleicht täte er gut daran, diese nicht nach dem Muster industrieller Technologie als beliebig formbare Herstellungstechnologie misszuverstehen; und doch stehen auch schon mit dem bereits vorhandenen Genmaterial unglaubliche Optionen des Gestaltens von biogenen Prozessen bereit. Die Gratisgabe der Reproduktion ist all diesen Agentien immanent – artifizielle Ste-

2 Auch der Wikipedia-Artikel zu Open Source lässt hier einen gewissen Interpretationsspielraum offen:

> „Der Begriff Open Source beschränkt sich nicht ausschließlich auf Software, sondern wird auch auf Wissen und Information allgemein ausgedehnt. Beispiele dafür sind *OpenCola* und auch *Wikipedia* selbst. Übertragen wurde die Idee des öffentlichen und freien Zugangs zu Information auch auf Entwicklungsprojekte. In diesem Zusammenhang wird dann oft von Open Hardware gesprochen, wobei es sich nicht um freien Zugang zur Hardware handelt, sondern um freien Zugang zu allen Informationen, eine entsprechende Hardware herzustellen." (http://de.wikipedia.org/wiki/Open_Source)

rilität durch Terminatorgene einzubauen war oft eine der ersten Taten derjenigen, die auch Naturprozessen eine proprietäre Verwertbarkeit einprägen wollen.[3]

Die Auseinandersetzung um die immer stärkere Einschränkung des naturwüchsigen freien Umgangs mit genetischem Material[4] kann sich in ihrer Schärfe locker mit der um Softwarepatente messen; im landwirtschaftlichen Bereich zählt dazu vor allem die Frage des sogenannten Bauernprivilegs, also die Erlaubnis für den Bauern, auch sortengeschütztes Erntegut des Vorjahres auf den eigenen Feldern im Folgejahr wieder auszubringen (Nachbaumöglichkeit) und sich auf diese Weise vom Zwang, jährlich neues Saatgut zukaufen zu müssen, unabhängig zu machen. In Österreich ist dieses sogar mit einem Anrecht auf eine geringe Menge auf jeweils neuestes Saatgut aus staatlichen Versuchsanstalten verbunden.

Ich habe meine Reflexionen über die der Open Source entsprechende materielle Produktionsweise mit Absicht mit einem Beispiel aus der Biologie und aus dem ländlichen Raum begonnen, denn der Sinn dieses Aufsatzes ist es, einen spezifischen Ansatz vorzustellen, unter dessen Bedingungen Open-Source-Prinzipien in der materiellen Produktion verwirklicht werden können.

Erstes Kriterium: Zugang zu Ressourcen Materielle Produktion ist gebunden an das Vorhandensein von Rohmaterialien, von Stoff und Stoffumsatz, und die Perspektive eines substantiellen und wirksamen Interesses von Produzenten an einem freien Austausch von Produktionswissen hängt sehr damit zusammen, dass auch die übrigen Kosten der Produktion die Eigenproduktion für eigenes oder fremdes Bedürfnis wieder attraktiv machen – und das setzt einen freien Zugriff auf stoffliche Ressourcen voraus. Eine der wesentlichen Argumentationen für die

3 Siehe hierzu Hänggi (2006) und ihre Reflexionen über Richard Jefferson, der in Sachen Gentechnologie glühender Befürworter, in Sachen Patentierbarkeit entschiedener Gegner der Biokonzerne ist.

4 Hierzu Hänggi:

> „Jefferson gründete 1992 das unabhängige Forschungsinstitut Cambia und wurde zum Vater der Open-Source-Bewegung in der Biotechnologie, indem er die Bios-Lizenz entwickelte (*Bios* steht für *Biological Innovation for an Open Society*). Open Source bedeutet: Jede und jeder darf ein 'geistiges Eigentum' gratis nutzen und weiterentwickeln, wenn er oder sie die Weiterentwicklungen wiederum gratis weitergibt. [. . .]
>
> Freilich: So einfach ist das 'Verschenken' biologischer Errungenschaften nicht. Man könnte einfach auf ein Patent verzichten. Das birgt aber erstens die Gefahr, dass ein anderer das Patent beantragt. Das wäre zwar Piraterie, aber wenn niemand ein solches missbräuchliches Patent anfechtet, gilt dieses. Zweitens hat, wer auf ein Patent verzichtet, keinen Einfluss darauf, wie seine Errungenschaft verwendet wird. Deshalb tut Cambia das, wovon die Organisation eigentlich findet, es dürfe gar nicht möglich sein: Es lässt biologische *Erfindungen* patentieren – um sie dann gratis zu lizenzieren."

Und vorher bemerkt Hänggi: „Cambia kämpft mit der Bios-Lizenz gegen *Monsanto*, *Dupont*, *Syngenta* oder *BASF*, indem es biotechnologische Werkzeuge entwickelt – und 'verschenkt'. Beispielsweise *Trans Bacter*: Dieses Bakterium kann fremde Gene in ein Zielgenom einbringen. In der Regel wird zu diesem Zweck das Agrobakterium verwendet, das aber mit mehreren Patenten belegt ist; wer es benutzt, muss Lizenzgebühren zahlen. *Trans Bacter* ist gratis."

Dominanz proprietärer Produktionsformen (und das heißt heute im Wesentlichen Monopolisierung von Produktionswissen und Verfahren) ist wie bereits erwähnt, dass der Zugang zu materiellen Produktionsbedingungen ungleich schwieriger zu haben ist als der zu geistigen. Die Frage ist, ob eine Forcierung und Diversifizierung erneuerbarer Ressourcen zu einer Veränderung dieser Situation führen kann.

Zweites Kriterium: Fähigkeit zu produzieren Open Source als Organisationsweise geistiger Arbeit hat sich in den Bereichen Software und Wissensproduktion deswegen durchgesetzt, weil das Produktionsmittel Computer massenhaft und billig zur Verfügung steht, weil der Rohstoff Code sich blitzartig kopieren und reproduzieren lässt und weil auch die Kommunikation zwischen kooperierenden Produzenten derart revolutioniert wurde, dass sie nicht mehr aufwändig an einem Ort versammelt werden müssen. All dies sei im Bereich der materiellen Produktion nicht gegeben. Produktion sei gebunden an die Zurverfügungstellung eines gewaltigen Kapitalvorschusses, unter dessen Kontrolle und Regie sie am effizientesten ablaufen würde. Die *economies of scale* und die beständige Verdichtung der Märkte und Produktionslogistiken sorge außerdem dafür, dass sich der „Eintrittspreis" für konkurrenzfähige Produktion beständig erhöhe. Die Frage ist, ob auch diese Logik prinzipiell unterlaufen werden kann, zum Beispiel durch Dezentralisierung der Produktion und Förderung der Eigenarbeit.[5]

3 Die biomorphe Wende der Technologie und die kybernetische Revolution der Biologie

Die technische Entwicklung ist tatsächlich dabei, in beiden Bereichen gültige Verhältnisse zu überwinden. Die Entwicklung der computergesteuerten Automation hat nicht nur dazu geführt, dass menschliche Arbeitskraft im industriellen Produktionsprozess durch Maschinen ersetzt wurde; sie hat vor allem dazu geführt, dass sich

5 Sehr skeptisch äußert sich in diesem Zusammenhang Marianne Groenemeyer (2004):

> „In einem studentischen Projekt, das sich über zwei Semester erstreckte, sind wir intensiv der Frage nachgegangen, welche Möglichkeiten zur Eigentätigkeit und zur Minderung des Geldbedarfs es in den Bereichen Nahrung, Kleidung, Wohnung und Bildung in den modernen Gesellschaften gibt. Das deprimierende Ergebnis unserer Nachforschungen: Die konsumistische Gesellschaft hat die beiden in ihr favorisierten Existenzweisen, nämlich Warenproduktion und Warenkonsumtion, so totalisiert, dass beinah jede andere nicht von solcher Produktion und solchem Konsum beherrschte Tätigkeit erstorben ist. Nicht zuletzt dadurch, dass es schlichtweg kaum noch Eigenarbeit gibt, die ihren Einsatz lohnte. Jede Eigenarbeit wird durch Billigprodukte von vornherein ins Unrecht gesetzt oder entmutigt."

Dies gibt nur eine relative, aber keine absolute Bedingung an: Eigenarbeit muss wesentlich attraktiver und produktiver werden, als sie heute gestaltet ist.

mehr denn je die Produkte dieses industriellen Produktionsprozesses selbst als flexible Produktionsmittel im Kleinen gebrauchen lassen. „Die Entwicklung" ist dabei ein höchst unvorhersehbares System von kostenersparender Arbeitsexternalisierung und konkurrenzbedingten Effizienzzwängen, die bis in das Herz der privaten Haushalte hinein das Leben verändern. Die beständige Verkleinerung und Verbilligung elektronischer Bauteile bringt es mit sich, dass immer mehr Steuerungsintelligenz in immer mehr Produkten „inkorporiert" werden kann.

Alvin Toffler und Marshall McLuhan haben diese Entwicklung schon früh erkannt und eine Art Selbstaufhebung der Industriegesellschaft vorausgesagt: Marshall McLuhan (1962) prägte den Begriff des *Global Village*, Toffler und Toffler (2006) haben den Ausdruck des *prosuming* geprägt, also die Verwischung der Grenzen zwischen Produktion und Konsumtion. Die industriell produzierten Automaten ermöglichen in immer mehr Domänen zwischen Produktion und Konsum eine Steigerung der Eigentätigkeit und bringen allmählich einen neuen sozialen Charakter hervor, der nicht mehr in die klassische Dichotomie von Angestellten und Selbständigen passt. McLuhan hat vor allem den Aspekt der Dezentralisierung in den Vordergrund gestellt, also die Aufhebung der Fabrik und der industriellen Stadt in ein Netzwerk von Informations- und Materialflüssen.

Was die Natur in Form von Leben schon längst hervorgebracht hat, das perfekte dezentrale autonome Funktionieren komplexer Systeme, die sich selbst erhalten, reparieren, gruppieren, das implementiert nun auch die menschliche Technologie langsam und schrittweise. Kevin Kelly hat in seinem Buch „Out of Control" diesen biomorphen Grundzug der Technikentwicklung dargestellt (Kelly 1995). Stephen Wolfram versucht in seinem Buch „A New Kind of Science" eine theoretische Grundlegung „zellulärer Automaten" (Wolfram 2002).

Auf der Rückseite dieser Entwicklung spielt sich aber eine ebenso faszinierende Parallelentwicklung ab; eine verfeinerte Auffassung der Natur, die diese nicht mehr als mystisches und geheimnisvolles Jenseits des Menschlichen begreift und auch nicht als triviale Anhäufung zufälliger Mutationen, in die der Mensch beliebig eingreift, sondern als einen unendlich komplexen und lernfähigen Systemzusammenhang. Autoren wie John Lyle und John Todd haben Pionierarbeit geleistet bei der Aufbereitung des Verständnisses natürlicher Systeme als fein abgestimmte kybernetische Regelkreise, in denen sich die Evolution von Metasystemen und die Evolution von Organismen gegenseitig unterstützen.[6] Dieses Verständnis der Natur als hochkomplexer kybernetischer Prozess, wie es sich in Termini wie „Lebende Maschine" ausdrückt, hat in den letzten Jahren zu einem Aufschwung von Ansätzen geführt, die nicht einzelne Organismen kultivieren, sondern Metasysteme, also vom Menschen bewusst eingesetzte komplette Biotope. Ein Beispiel sind Biotope, die Wasser weit kostengünstiger reinigen als herkömmliche Filter, sich im Zuge dieses Prozesses ständig selbst erneuern und als Draufgabe noch Nahrung und Energie liefern. Der bekannteste dieser

6 Vergleiche Todd und Todd (1994) sowie Lyle (1996) zur Einführung in die Arbeit dieser Autoren.

Ansätze ist die Permakultur[7], und er zeigt gegenüber herkömmlicher Landwirtschaft erstaunliche Überlegenheit in Sachen Effizienz und Nachhaltigkeit von Prozessen auf. Auf einer ganz anderen Ebene zeigt Lynn Margulis, dass der menschliche Organismus selbst ein komplexes Symbiosesystem mit anderen lebenden Organismen ist und diese Symbiose gelenkt werden kann (Margulis und Case 2006).

Auf der Grundlage dieser doppelten biologisch-kybernetischen Denkrevolution hat sich eine Fülle von synthetisierenden Sichtweisen entwickelt, die zeigen, dass menschliche Produktion anders und vernünftiger organisiert werden kann, wenn sie sich insgesamt als regeneratives System auf der Basis evolutionär bewährter oder auch neuartiger Muster[8] organisieren lässt. Der Stoffdurchsatz und die Reproduktionsrate lassen sich zwar nicht beliebig steigern, wie dies die industrielle Denkweise anzunehmen scheint, aber dafür ist es möglich, relativ hohe Level einer Homöostase von Systemen zu erreichen, wenn sie richtig – und das heißt unter Bedachtnahme auf sämtlichen involvierten Ökosysteme – gestaltet sind.

Wenn man bedenkt, dass unter herkömmlichen Bedingungen im globalen Maßstab noch immer kulturlandschaftlich genutzte Landfläche in Dimensionen von zigtausenden Quadratkilometern jährlich verloren geht, so erscheint die Alternative einer forciert auf die Wiederaneignung solcher Reproduktionsräume gerichteten Politik und Ökonomie nicht ganz abwegig. Ein wichtiger Aspekt im planetaren Management ist die gegenwärtige Überfülle an freiem atmosphärischen CO_2, die aus der jahrzehntelangen Umwandlung biogener Reststoffe wie Erdöl resultiert.

Eine Verbindung von biomorpher Technologie mit der freien Energie der Sonne erscheint als eine mögliche Option, auch im stofflichen Bereich jene Nicht-Knappheit zu erreichen, die das Aufblühen der freien Wissenswelt ermöglicht hat. Genau hier ist auch eine jener doch erwähnenswerten Einzeltechnologien angesiedelt, die sich heute schon unter die Verfahrensweise Open Source im stofflichen Bereich einordnet: die Solar-Roof-Technologie von Rick Nelson. Dabei geht es um „Blau-Grüne" Dächer mit transparenten Wärmespeichern voller Algen und Wasser, die CO_2 in Biomasse sequestrieren und so Energie speichern.[9]

An die Stelle eines direkten evolutionären Kampfes oder eines sanften Verdrängungswettbewerbs zwischen „Homo Carbonis" und „Homo Siliconis", wie sie in dystopischen Bildern von Paolo Soleri[10] bis Hans Moravec[11] immer wieder beschworen werden und in Filmen wie *Matrix* und *Terminator* eine eindrucksvolle Bebilderung erfahren haben, tritt die Vision einer Kooperation von menschgemachten und natür-

7 Der Begiff Permakultur wurde von Bill Mollison und David Homgren geprägt, Näheres etwa bei http://www.permakultur.de.

8 Unter http://www.patternlanguage.com definiert Christopher Alexander *Muster* als bewährte Strukturen, die immer wiederkehrende Aktivitäten unterstützen und so ein Bindeglied zwischen konservativem und imaginativem Denken schaffen, das der Moderne weitgehend verloren gegangen sei.

9 http://www.solaroof.org

10 Weiterführend hierzu http://www.arcosanti.org.

11 Siehe etwa Moravec (1999).

lichen Prozessen, die in immer steigender Verfeinerung einander unterstützen und bedingen.

4 Die Idee der globalen Dörfer . . .

Nach diesem groben und kursorischen Exkurs kann der philosophische Standpunkt, den ich mit „Idee der Globalen Dörfer" bezeichne, in seinem Fundament kenntlich gemacht werden. Globale Dörfer sind Lebens-, Wohn-, Arbeits- und Technikformen, die auf beiden Säulen aufbauen, der technologischen und der ökologischen, die sozusagen am perspektivischen Schnittpunkt der Leistungen beider Teilsysteme angesiedelt sind. Die ungeheuren Möglichkeiten der Technologie sollen mit den imposanten Möglichkeiten der vom Menschen verstandenen, optimierten und mit der menschlichen Existenz synthetisierten „natürlichen", sprich: biogenen Metasysteme kombiniert werden. Beide Subsysteme sollen sich weiter aneinander annähern, ohne sich in ihrer spezifischen Wirkungsweise wechselseitig aufzuheben, wie zum Beispiel ein technizistischer Eingriff in die Natur es ohne jeden Zweifel tut.

Globale Dörfer gehen von der Überzeugung aus, dass dezentrale und lokale Prozesse in noch nicht absehbarer Weise verfeinert und effektiviert, kombiniert und diversifiziert werden können. Primäre Voraussetzung dafür sind Wissen und Design. Es geht daher um eine Siedlungs- und Lebensweise, die global generiertes Wissen und Design mit lokaler Wirksamkeit und Anwendbarkeit dieses Wissens verbindet.

Damit wäre auch eine dritte Komponente der auf Open-Source-Prinzipien basierenden Produktionsweise gegeben: Wie Peter Fleissner an dieser Stelle vor einem Jahr treffend bemerkte, hat Open-Source-Entwicklung eine „Achillesferse", nämlich den Umstand, dass dabei viel unbezahlte Zeit aufgewendet werden muss (Fleissner 2006). Dieses Argument wiegt umso schwerer, als mit steigender Komplexität einer Technologie die zu ihrer Entwicklung und Anpassung benötigte Zeit exponentiell zunimmt. Zwar ist es richtig, dass gerade Open-Source-Technologien ein hohes Verlässlichkeitspotenzial haben und daher nicht der beständigen Aufmerksamkeit des Benutzers bedürfen. Dennoch ist gerade im Kontext sich ständig wandelnder Lebenssituationen, der beständigen Interaktion von Prozessen und der dynamischen Weiterentwicklung von Benutzerbedürfnissen ein geradezu endemisches Problem von Open Source, dass zu wenige Ressourcen für ihre Weiterentwicklung gegeben sind. Es darf vermutet werden, dass es mit freien Designs, Automaten und Bauplänen nicht anders bestellt wäre.

Während Peter Fleissner dieses Problem über Grundeinkommen und Verwertungsgesellschaften zu lösen versucht, geht die Idee der globalen Dörfer von einem grundlegend anderen Ansatz aus. Sie behauptet, dass sich durch Wohlstandseffekte der Anwendung freier Technologien das Problem der materiellen Absicherung sozusagen von selbst erledigt, weil dadurch eine wesentliche Basis gemeinschaftlicher Selbstversorgung in Absprache und damit die Gewinnung disponibler Zeit verbunden ist. Die dazu notwendigen Überlegungen werden insbesondere im Kontext der Ideen

und Schule von Frithjof Bergmann verfolgt, der in Südafrika begonnen hat, solche um gemeinschaftliche Eigenversorgung auf Basis kybernetischer Technologien kreisenden Konzepte als Rezept gegen Armut und Arbeitslosigkeit zu realisieren.[12] Für die Metropolen mögen solche Konzepte derzeit utopischer denn je erscheinen, aber sie sind vielleicht dennoch realistischer als der Versuch einer Aktivierung sozialstaatlicher oder monetärer Mechanismen in einer Zeit, in der längst alle Möglichkeiten des simulativen Kreditwesens in Anspruch genommen werden, um den Zusammenhang von Staat und Markt weiter aufrechtzuerhalten.[13]

5 . . . und vier bis fünf Bausteine auf dem Weg dorthin

Soweit die Theorie. Nur – wo sind die globalen Dörfer? Ist diese Idee einer Verallgemeinerung dezentraler Siedlungsformen, ihrer Vernetzung und kooperativen Organisation, ihrem Technologieaustausch und ihrer zunehmenden Bedeutung für Wirtschaft und Gesellschaft weltweit nur eine schöne Vision,[14] ein utopisches Hirngespinnst oder bestenfalls ein netter Urlaubsort oder Zweitwohnsitz für Betuchte, wie Colletta di Castelbianco[15] in Ligurien? Tatsächlich gibt es erst Spurenelemente davon, so etwa in Kirchbach im österreichischen Bundesland Steiermark, wo einige junge Leute, inspiriert von der Idee der Globalen Dörfer, ein altes Gerichtsgebäude gekauft und eine Entwicklung in diese Richtung eingeleitet haben.[16]

5.1 Lokale Stätten des globalen Lernens

In den letzten Jahrzehnten konnte man eine noch nie dagewesene Konzentration von ökonomischen Aktivitäten und Bevölkerung in Ballungsgebieten beobachten; „Globalisierung" scheint gleichbedeutend mit „globaler Urbanisierung". Dies ist ein weltweiter Trend mit tiefgreifenden Auswirkungen auf Europa. Die gewerbliche Wirtschaft bildet Cluster, um Lieferanten und Verteilungskanäle nahe zusammenzubringen und „just in time" zu liefern. Ein hochkomplexes logistisches Netzwerk braucht Nähe zu Autobahnen und die Möglichkeit, jegliches Problem binnen Minuten zu lösen – ein „Ort der Flüsse" entsteht oft jenseits alter und historischer Zentren.

Regionale und ländliche Räume scheinen die Verlierer dieser Entwicklung zu sein. Qualifizierte Menschen werden angezogen durch urbane Gebiete, lokale Wirtschaftszweige sind in vielen Regionen am Aussterben, und noch immer folgt vielerorts der Zusammenbruch der lokalen Versorgungssysteme: Postämter, Geschäfte und Lokale schließen in ländlichen Gegenden, Staats- und Dienstleistungsinstitutionen bieten

12 http://www.neueArbeit-neueKultur.de
13 Weiterführend hierzu Kurz (1999) sowie Kurz (2003).
14 http://www.oekonux.de/texte/globdorf.html
15 http://www.collettadicastelbianco.com/en/
16 http://kb5.at/content/e1267/index_ger.html

ihre universellen Dienste nicht mehr länger an. Dies hat tiefgreifende Effekte auf die Lebensqualität sowie das Potenzial, eine Abwärtsspirale auszulösen.

Die Gegenstrategie, die in manchen Bereichen der europäischen Regionalentwicklung, aber auch in Ländern mit einem hohen Anteil an prekärer Infrastruktur, wie in Südafrika[17] oder Indien, existiert, hat zu einer Annäherung von Regionalentwicklungssystem und Bildungssystem geführt. Eine Idee eines auf lokale Umstände und Besonderheiten abgestimmten Bildungssystems trifft auf die Sachzwänge des faktischen Ressourcenmangels regionaler Körperschaften. Wesentlich ist, dass der Anspruch an diese Art von Bildung eine höhere Qualität erfordert als die traditionelle standardisierte Bildung. Gesucht wird eine Art von Bildung, die es Menschen ermöglicht, in ihrer Region zu bleiben, Gelegenheiten vor Ort zu finden und zu entwickeln und bestehende Ressourcen und Gelegenheiten kreativ zu nutzen – die sich von Situation zu Situation unterscheiden können.

Auf Internet und geteiltem Wissen aufbauende Bildungs- und Lernstätten auf Gemeindeebene und deren zunehmende Verknüpfung mit lokalen Entwicklungsfragen scheinen für dieses Problem eine geeignete Lösung zu sein. Sehr schön hat das schon 1994 der italienische Autor Giuseppe Silvi beschrieben, als er die Idee der „Piazza Telematicha" in die Welt setzte (Silvi 1999). Doch die Anzahl der simultanen Ideen ist fast endlos, daher nur zwei Verweise: Adolf Jändls Lernstatt in Bayern[18] und der Lernende Stadtteil.[19]

Dieses neue lokale Bildungssystem, das die Chance einer ausgeglichenen Entwicklung wahrzunehmen versucht, die permanent zwischen zahlenmäßig beschränkten AkteurInnen ausverhandelt werden muss, kann in mehrfacher Hinsicht von elektronischen Medien profitieren. Zum einen erlaubt das elektronische Medium die Koordination lokaler Aktivitäten zu verfeinern.[20] Zum anderen ist der Zugriff auf freie Bildungsressourcen auf der Basis einer globalen Open Source wahrscheinlich die einzige Chance, den Ansprüchen einer immer diffizileren lokalen Aufgabenstellung gerecht zu werden. In diesem Zusammenhang ist es bemerkenswert und wichtig zu sehen, wie zentrale Bildungsinstitutionen im eigenen Interesse (Reputation, Gewinnung von Studenten) zunehmend Lerninhalte freigeben. Neben dem bekannten Beispiel des *MIT* möchte ich hier insbesondere Angebote hervorheben, die der lokalen Umsetzung angepasst sind, etwa die *community toolbox* der Universität Kansas, USA, mit alleine 6 000 Seiten.[21]

17 „Multipurpose Community Centers are places where communities can hold forums to share information and lessons on successful development efforts." (http://www.gcis.gov.za/mpcc/about/whatare.htm)

18 http://www.ab-dat.de/lis/telecentrum/projekte/LernStatt.html

19 Gabriele Steffen gibt hierzu folgende Beschreibung: „Es gibt neue Modelle für Lernorte, die sich nicht nur am Modell des Klassenzimmers orientieren, sondern etwa auch an dem des Cafés (wie Internet- oder Philosophiecafés), des Kaminzimmers, des Salons, der Kneipe, der Werkstatt oder des Ladens." (http://www.die-bonn.de/efil/expertisen/steffen01_01.htm)

20 http://www.upmystreet.com aus Großbritannien zeigt sehr schön auf, wie lokale Diversität sichtbar und nutzbar gemacht werden kann.

21 http://ctb.ku.edu

Von hier ist es ein folgerichtiger, aber keineswegs einfacher Schritt, die Allianz der Zugangs- und Lernorte als Koordinierungsinstrument für Verfeinerung und Qualitätssteigerung von Inhalten zu fordern.[22] Die Verbindung von solchen Zugangs- und Lernorten mit Möglichkeiten der dezentralen Produktion und „Realisierung" von Bauplänen und Designs ist eine weitere logische Konsequenz, die allerdings selbst sehr viele Voraussetzungen hat, wie sich im Folgenden zeigt.

Eine wesentliche, technikimmanente Voraussetzung ist die Synchronizität der Prozesse. Eine neue Generation von Online-Lernwerkzeugen integriert atmosphärisches Video sowie verfeinerte Entwurfs- und Kommunikationswerkzeuge. In diesem Zusammenhang ist bemerkenswert, dass Markus Merz vom Open-Source-Car-Projekt *OScar*[23] die Verfügbarkeit von komplexen Tools der Kooperation zur *conditio sine qua non* für die Möglichkeit von offenen Designprozessen etwa von Automobilen gemacht hat, die sich dann wiederum in dezentralen Werkstätten realisieren lassen.

Das Projekt *Video Bridge*, an dem der Autor beteiligt ist, ist in diesem Zusammenhang zu sehen als ein Versuch, die Möglichkeiten der Breitbandkommunikation in die Stärkung lokaler Bildungshäuser umzusetzen.[24]

5.2 Metamorphosen der traditionellen „kleinbürgerlichen" Produzentenautonomie zu virtuellen genossenschaftlichen Verbünden

Eigentlich ist mit der immanenten Tendenz zum informationellen Monopol, die in mehr und mehr Sphären der Wirtschaft um sich greift, die materielle Grundlage für eine (quasi als letztes Überlebensmittel unausweichliche) Renaissance von Produzentennetzwerken gegeben. Ökonomische Rationalität der Kooperation macht sich daher in immer mehr Sphären der profanen Wirtschaft breit, gerade weil die Logik der Konkurrenz sich immerzu verschärft. Dabei kommt gerade der lokalen Kooperation eine entscheidende Rolle zu.[25]

22 Ein Überblick über Telecenter findet sich unter http://ebook.telecentre.org/pdf/telecentre_ebook.pdf, worin Telecenter einleitend beschrieben werden:

> „A telecentre is a public place where people can access computers, the Internet and other technologies that help them gather information and communicate with others at the same time as they develop digital skills. While each telecentre is different, the common focus is on the use of technologies to support community and social development—reducing isolation, bridging the digital divide, promoting health issues, creating economic opportunities and reaching out to youth. [...] Telecentres exist in almost every country on the planet, although they sometimes go by different names (e. g. village knowledge centres, infocentres, community technology centers, community multimedia or school-based telecenters)."

23 http://www.theoscarproject.org

24 Das Video-Bridge-Projekt strebt nach der Herstellung von permanenten und erschwinglichen Kommunikationsmöglichkeiten für Gruppen vor allem in ländlichen und peripheren Räumen durch atmosphärisch dichte Videoübertragungen (http://www.dorfwiki.org/wiki.cgi?VideoBridge).

25 Ein Beispiel für das gemeinsame „Kundenmanagement" von Kleinbetrieben in einer Region und den nahezu von den Umständen erzwungenen Zusammenschluss und die Überwindung der Konkurrenz unter den Kleinunternehmern als Ausgangspunkt sozialer Innovationen in regionalen ökonomischen

Während auf der einen Seite halbgeschlossene Wissensgenossenschaften wie der deutsche Schreinereiverbund *Koncraft*[26] entstehen, gelangen nunmehr selbst große Industriebetriebe schrittweise zu der Erkenntnis, dass sie nicht mehr genug wissen, um die komplexen Bedürfnisse ihrer Kunden zu erfassen. Die „Chinesische Mauer" zwischen der Produktion und der Konsumption öffnet sich also langsam, um das Bedarfswissen der Kunden in der Produktion nutzbar zu machen. Wenn Kunden jedoch nicht ihrerseits über Wissen über prozessuale Bedingungen der Produktion verfügen, vermögen sie keine Inputs zu geben. „Co-Design-Software" ist vielleicht der erste Schritt der (Wieder-)Vergesellschaftung des Produktionswissens (Reichwald und Piller 2006).[27]

Mittlerweile hat in der akademischen Welt die Anerkennung der Faktizität dieser grundlegend neuen Phänomene weite Kreise gezogen. Eine Pionierfunktion kam in diesem Zusammenhang der Harvard-Professorin Shoshanna Zuboff zu, die in ihrem Buch „The Support Economy" die simultane Veränderung sowohl der betrieblichen als auch der individuellen Qualifikationsstruktur beleuchtet und von einer „kopernikanischen Wende" spricht, in der Individuen mit einem wesentlich erweiterten Möglichkeitenbewusstsein die Gestaltung des eigenen Lebens in die Hand nehmen (Zuboff und Maxmin 2004). Es stellt sich die Frage, inwieweit dies nicht auch auf Gruppen von Individuen zutrifft, die aus gemeinsamen Impulsen und Werthaltungen heraus auch einen gemeinsamen physischen Raum ihrer Entfaltung suchen.[28] Es ist sicher nicht übertrieben, vom Aufstieg einer Kultur der Partizipation zu sprechen, die sich heute vor allem um Mittel und Werkzeuge gruppiert, die mit dem Begriff Web 2.0 beschrieben werden.[29] Yochai Benkler (2006) prägte in diesem Zusammenhang den Terminus der *commons-based peer production*.

Eine internationale akademische Arbeitsgruppe mit TeilnehmerInnen aus Großbritannien, Brasilien und Südafrika erforscht derzeit unternehmerische Konzepte, die von vorneherein auf offenem Wissen aufgebaut sind.[30] Eine sehr breit angelegte Bemühung, diese Innovationen zu erfassen und zu unterstützen ist die *Foundation for Peer2Peer Alternatives*, die die bis heute umfangreichste Sammlung von kooperativen Projekten vorgelegt hat.[31] Eine Unzahl ähnlicher und befreundeter Projekte existiert, wobei vermutlich Zusammenschlüsse und Verschiebungen bevorstehen. Stell-

---

Systemen gibt *wedge* (http://www.wedgecard.co.uk).

26 http://www.koncraft.de

27 Siehe auch den Beitrag „Interaktive Wertschöpfung – Produktion nach Open-Source-Prinzipien" von Frank Piller, Ralf Reichwald und Christopher Ihl auf Seite 87 in diesem Buch.

28 http://www.ic.org

29 „The Rise of The Participation Culture" bietet eine gute Einführung ins Web 2.0: http://www.wsjb.com/RPC/V1/Home.html.

30 Unter http://www.openbusiness.cc/category/models findet sich die meines Erachtens derzeit beste verfügbare Fallsammlung von Unternehmenskonzepten. Seit August 2005 werden dort monatlich mehrere Fallstudien präsentiert.

31 http://p2pfoundation.net/Category:Business und http://p2pfoundation.net/Main_Page

vertretend sollen nur drei erwähnt werden: The Transitioner[32], Howard Rheingolds *Cooperation Commons*[33] sowie das *Meta Collab*[34] in der Wikia-Welt. Es gibt auch kooperative Netzwerke, die dem Thema der globalen Dörfer besonders nahe stehen: Frithjof Bergmanns schon erwähntes Netzwerk der Neuen Arbeit, das sich insbesondere der Wiederaneignung von Selbständigkeit und Eigenmacht durch die von der Erwerbsgesellschaft Marginalisierten widmet,[35] während Christine Ax (2002) mit ihrer Initiative „Handwerk der Zukunft" zeigt, dass die traditionellerweise unverbundenen Kleinbetriebe auch und gerade in ländlichen Räumen in kooperativen Netzwerken der Produktion und des Wissens eine Zukunft haben:

> „Handwerk kooperiert und dies auf viele verschiedene Art und Weisen. Die 'gut geführte Flotte' als Alternative zum Tanker ist Realität. Das Handwerk der Zukunft ist die Zukunft des Handwerks und findet heute bereits statt."[36]

Ein kritischer Punkt ist die Frage, ob sich von monetären Kreisläufen verschiedene Formen des realen und effektiven Austauschs von Gütern in die Tat umsetzen lassen. Angefangen bei der elektronischen Abwicklung von Tauschkeisen[37] über die anspruchsvolle PlaNet-OpenCoop-Initiative[38], die versucht, nachhaltigkeitsrelevante und gemeinwohlorientierte lokale soziale Unternehmungen weltweit zu vernetzen, bis hin zu Ansätzen eines nichtmonetären Netzes des freien Austauschs von kreativen Energien und Produkten reichen die Ansätze.[39]

32 Ein Wiki mit folgender Selbstbeschreibung: „TheTransitioner.org brings together those who want to develop Collective Intelligence in order to build a fair world" (http://www.thetransitioner.org).

33 Eine Zusammenarbeit zwischen dem *Institute for the Future* und Howard Rheingold: http://www.cooperationcommons.com.

34 http://collaboration.wikia.com

35 http://www.newwork-newculture.de

36 So beschreibt es der Text auf der Leitseite des Instituts für Produktdauer-Forschung in Hamburg (http://www.ipf-hamburg.net).

37 Üblicherweise erstellen Tauschkreise ein nach Rubriken geordnetes Verzeichnis mit allen Angeboten und Gesuchen der Mitglieder, das regelmäßig aktualisiert wird. Wann, wo und auf welche Weise eine angebotene Tätigkeit ausgeführt wird, vereinbaren die jeweils Beteiligten. Aus: http://de.wikipedia.org/wiki/Tauschkreis.

38 Die Initiative beschreibt sich selbst wie folgt:

> „An 'Open Organisation' is a functional structure for people to work together. It is open to anyone who agrees to abide by its purpose and principles, with complete transparency and clearly defined decision making structures, ownership patterns, and exchange mechanisms; designed, defined, and refined by all members as part of a continual transformative process. Our current projects are guided by these principles. We see them as the first steps towards a global network of self-organised Local Open Co-ops which we hope will eventually become entirely self funding as an ethical trading network with unique features including reputation and trust feedback mechanisms, flexible (user-definable) group-based policy/decision making rules, and complementary barter and micro-credit currencies." (http://open.coop/principles.htm)

39 Eine Initiative beschreibt beispielhaft die Grundidee:

5.3 Die Dezentralisierung der Produktionsmittel

In einer Zeit, in der sich durch die Mikroelektronik und Automationstechnik die Möglichkeit ergibt, immer versatilere Werkzeugmaschinen herzustellen, häufen sich auch die Vorschläge, wie dem Dilemma einer unkontrollierten und verschwenderischen „wirtschaftlichen" (schon das Wort hat sich bedeutungsmäßig ins Gegenteil verkehrt) Wachstumsdynamik entgegenzutreten ist. Die allgemeine Technologieentwicklung macht niederskalare effektive Produktion möglich (Automation als sinnvollste Extraktion von Routinen und eine kleinen Gruppen und geringem Verbrauch angemessene Technik, für möglichst spontane absatzunabhängige Produktion).

Die im vorigen Abschnitt erwähnten Produzentennetzwerke basieren zwar derzeit noch primär auf dem Austausch von Wissen und Information, doch wäre schon dies ohne dezentrale Produktionsmittel undenkbar, die diese Information in Bearbeitung von Stoffen umsetzen. Diese aber generieren einen immer größeren Bedarf an flexibler Produktionsintelligenz, die in Repositorien und Produktteildatenbanken auf „virtuell-genossenschaftlicher" Grundlage erfasst werden könnte.

Die derzeit sehr stark vom *personal fabricator*[40] dominierte Debatte verdeckt vielleicht sogar das Ausmaß der kleinräumigen Kombination von Computern und Werkzeugmaschinen, von Feldbussen, Erkennungs-, Sensor- und Messsystemen, CNC–Brücken, Laser-Cuttern und hunderten anderen innovativen Bearbeitungsverfahren. Ein Video des EU-Projektes *SMErobot*[41] erklärt sehr anschaulich, wie in einem lokalen Umfeld mit niedrigen Stückzahlen produziert werden kann. Diese Technologien sind vielleicht keine „personal"-Technologien, werden aber in einem „near-personal"-Umfeld zur Verfügung gestellt und können dort eingesetzt werden. Es lässt sich logischerweise gar kein besseres Absatzfeld als kundennahe Produktion im regionalen Umfeld denken, denn mit diesen Werkzeugen wird natürlich der Konkurrenzkampf am Zulieferermarkt zu den Industrien noch einmal angeheizt und dieser Markt für die meisten Firmen zunehmend unattraktiv.

Ich möchte hier Aussagen aus einem Spiegel-Interview[42] mit Tim O'Reilly wiedergeben, der die beschriebene Tendenz sehr schön auf den Punkt bringt. Die Welten der Eigenarbeit, der kleinen Fertigung, der regionalen Wirtschaft und globalen Netze

> „Give Get Nation is the world's first free, post-scarcity, values-based, person-to-person economy. We organize the world's surplus capacity and make it accessible to everyone for free. We're the world's first surplus product, labor, intelligence and spirit economy. Nothing here is for sale, barter or trade. There are no middle men, fees or overhead charges of any kind. Give Get Nation is an innovative social enterprise designed to improve the quality of life for everyone, set the world free from scarcity and save the environment through synergy." (http://www.givegetnation.net)

40 Weiterführend hierzu: die Website des *Fab Central* des *Massachusetts Institute of Technology (MIT)* http://fab.cba.mit.edu sowie Gershenfeld (2005).

41 http://www.smerobot.org/download/videos/SMErobot_Coffee-Break_400kps_DE.mov

42 Der Text findet sich bei *Spiegel Online* unter http://www.spiegel.de/netzwelt/web/0,1518,451248,00.html.

gehen auch in seiner Perspektive eine evolutionäre Synthese ein. Vielleicht ließe sich das, was daraus resultiert, noch am ehesten als „professionell unterstützte Eigenproduktion" beschreiben:

> „Unser nächstes großes Projekt steckt in einem neuen Magazin, das wir herausbringen. Es heißt *MAKE*. Wir konzentrieren uns darauf, wie Computer beginnen, mit der physikalischen Welt zu interagieren – wie beim Custom Manufacturing, der Maßanfertigung von Produkten. Gerade ist zu beobachten, dass viele Hacker Dinge herstellen: Die Leute haben ihre dritte oder vierte Digitalkamera, und was machen sie mit der alten? Sie können sie für Basteleien wiederverwenden. Inzwischen gibt es alle möglichen Geräte für derlei Maßproduktion, Laser-Schneidegeräte und 3D-Drucker, und sie kosten heute etwa so viel wie eine Schriftsatz-Maschine zur Zeit der Desktop-Publishing-Revolution. [. . .]
>
> Wir treten ein in das Zeitalter der Maßfertigung. Nehmen Sie den Bereich der synthetischen Biologie, wo sich diese Maßfertigung bis hinunter auf die Ebene der chemischen Prozesse und Stoffe erstrecken wird. Ganz zu schweigen davon, dass wir heute Dinge in relativ kleiner Stückzahl herstellen lassen können, in Ländern mit relativ niedrigen Lohnkosten. Es gibt immer ausgefeiltere Simulations- und Entwurfwerkzeuge. Leute konstruieren Dinge in *Second Life* oder mit *Sketchup* von Google. Solche Werkzeuge zur Gestaltung virtueller Objekte sind heute viel stärker demokratisiert, viel weiter verbreitet als je zuvor. Es gibt zum Beispiel Dienste, über die man sich seinen Avatar aus *Second Life* als 3D-Modell ausdrucken lassen kann. Das ist ein Aufbruch in neue Welten."

Übrigens: Auch schon das nach streng marktwirtschaftlichen und intellektuellen Eigentumsstandards konzipierte Second-Life-Universum ist schon damit konfrontiert worden, dass es oft sinnvoll ist, Designs zu kopieren und zu teilen anstatt sie zu verkaufen. Ein schönes Labor für eine Welt, in der die Ökonomie der (künstlich hergestellten) Knappheit nicht mehr gilt.[43]

5.4 Die Kombination von regenerativem Wissen mit regenerativen Ressourcen

Wenngleich auch die Vernetzung der Produzenten eine wesentliche Vorbedingung für die „Open Source der Güter" abgibt und die dezentrale Automation die Herstellung in ein nachgeordnetes Verhältnis zur Konzeption setzt, so ist doch noch immer die Frage, wie das Problem materieller Knappheit von Stoffen und Ressourcen nun konkret und tatsächlich gelöst werden soll. Wir erinnern uns an die eingangs zitierte Skepsis, und

43 Ergänzend zu Second-Life-Designs die Notiz auf dem Ticker des Heise-Verlags (http://www.heise.de/newsticker/meldung/81088) und die Diskussion auf keimform.de unter http://www.keimform.de/2006/11/16/kuenstliche-knappheit-in-second-life-bedroht.

auch an diesem Punkt arbeitet eine Fülle von Gruppen und Schulen an Modellen einer grundsätzlichen Veränderung.

Die Arbeit einer ganzen Reihe von Pionieren war notwendig, um so etwas wie globale Dörfer denkbar zu machen. Ihr Ausgangspunkt waren nicht traditionelle, in Bevölkerungszahl und Produktivität äußerst begrenzte Verfahren der materiellen Produktion und Lebensgestaltung, sondern eine völlige Neukonzeption von menschlichen Lebenswelten auf der Basis unseres vertieften Wissens über lebende Systeme und Naturprozesse. Es ist sinnlos, von Knappheit zu sprechen, denn diese Systeme sind auf beständigen Umsatz von Materie und Energie aufgebaut und lassen sich für viele Zwecke konstruieren. Nahrung, Energie, Wohnraum und vieles andere sind „Gratisgaben" eines intelligenten öko-logischen Designs.

An erster Stelle möchte ich die bahnbrechende Arbeit des verstorbenen John Lyle würdigen, der an der *California State Polytechnic University* in Pomona ein Zentrum für regeneratives Design aufgebaut hat.[44] Lyle gab sich nicht zufrieden damit, Lehrbücher für Studenten zu schreiben und einzelne Verfahren zu überprüfen, sondern konzentrierte sich auf synergetische Effekte zwischen architektonischen Interventionen und Naturprozessen. Auf der anderen Seite des Kontinents hat der nicht minder legendäre John Todd mit *New Alchemy* und *Ocean Arks* Laboratorien für geschlossene Stoffkreisläufe geschaffen, die den Menschen mit allem Lebensnotwendigem versorgen und sich weitgehend automatisch – eben als „lebendige Maschinen" – wiederherstellen.[45]

Technologisch und wissenschaftlich wurden in der Zwischenzeit viele weitere Schritte über diese Anfänge hinaus gemacht. Amory Lovins mit seinem *Rocky Mountains Institute*[46] ist in vielen Bereichen erneuerbarer Ressourcen tätig und wurde unter anderem bekannt, als er 1994 mit dem *Hyper Car*[47] das erste Mal eine komplexe technische Hardwarestudie bewusst in die Public Domain stellte. Lovins begründete so die Affinität zwischen Open Source und ökologischen Technologien deutlich mit, obwohl – vielleicht auch weil – er ein leidenschaftlicher Anhänger der Marktwirtschaft ist.[48] Weniger Open-Source-affin, dafür in ihren Konsequenzen ebenso revolutionär sind die Arbeiten von Michael Braungart und William McDonough,[49] die eine flächendeckende Rückkehr zu einer abfallfreien Technologie von regenerativen Ressourcen zu ihrem Programm gemacht haben und eine Fülle von einzelnen technologischen Ideen hervorgebracht haben. Ähnliches gilt für den in Wien wirkenden Chemiker Hanswerner Mackwitz, der die Leistungsfähigkeit einer auf Pflanzen als Ausgangsmaterial basierenden Chemie und Werkstofftechnologie propagiert.[50]

44 http://www.csupomona.edu/~crs/
45 http://www.oceanarks.org/links
46 http://www.rmi.org
47 http://www.rmi.org/sitepages/pid386.php
48 http://www.rmi.org/sitepages/pid1202.php, http://www.natcap.org
49 http://www.mbdc.com/c2c_nir.htm
50 http://www.alchemia-nova.net

Das *Institute for Local Self-Reliance* mit seinem Projekt *Carbohydrate Economy* ist nur einer der unzähligen Wissenspools, die aus diesen und ähnlichen Arbeiten resultieren.[51] Zu nennen wäre auch das *Institute for Sustainable Communities*[52], die *Schumacher Society*[53], das *Global Ecovillage Network*[54] und das *Sustainable Village*[55]. Den Schritt in die bewusste Übertragung von Open-Source-Prinzipien hat *Open Source Ecology* gemacht, eine Gründung des jungen amerikanischen Physikers Marcin Jakubowski.[56] In Osborn (Missouri)[57] begann die Entwicklung zum ersten dezidierten „Open Source Dorf" der Welt, leider mit dem Rückschlag, dass die Örtlichkeit im Jahr 2006 gewechselt werden musste. Doch gibt es mittlerweile ein neues, größeres Gelände. Auf diesem Gelände wird gelebt und gebaut, gelehrt und geforscht; alle Prozeduren werden dokumentiert und im Internet veröffentlicht. Ein schönes Beispiel ist der „hydroponische Salat".[58] Sämtliche ökologischen und monetären Kosten sollen transparent werden, wenngleich dieses Ziel noch lange nicht erreicht ist.

Open Source Ecology soll als lernendes Unternehmen selbstreplizierend wachsen, es werden sogenannte *Fellows* für zwei Jahre aufgenommen, die dafür eine Gebühr bezahlen; gleichzeitig wird ein monetäres Einkommen durch Lieferungen an eine sympathisierende urbane Kundschaft (*community supported agriculture*) bezogen. Zudem wird landwirtschaftliche Technologie von Grund auf so konzipiert, dass sie überall nachgebaut, repariert, eingesetzt und rezykliert werden kann.[59] Für diese modulare Technologie nach Baukastenprinzip wurde eine eigene Entwurfs- und Beschreibungssprache geschaffen.

Selbstverständlich ist es ein Ding der Unmöglichkeit, die vielen technologischen Innovationen, die ein globales Dorf benötigt, auch nur an einem Ort andenken zu wollen. Wenngleich auch jeder dieser Orte eine hohe interne Diversität aufweisen muss, so wird es ein ganzes Netzwerk von solchen Dörfern benötigen, um auch nur die elementarsten Schritte in Richtung einer nennenswerten Entwicklung zu machen. Vieles wird davon abhängen, ob es gelingt, mithilfe der Informations- und Kommunikationstechnologien die Entwicklungsanstrengungen vieler Dörfer – ich denke da nicht zuletzt auch an weiterentwickelte Fallbeispiele wie die energieautarken Dörfer und Städte in Mitteleuropa[60] – auf einen Nenner zu bringen.[61]

51 http://www.ilsr.org, http://www.carbohydrateeconomy.org
52 http://www.iscvt.org
53 http://www.smallisbeautiful.org
54 http://gen.ecovillage.org
55 http://www.thesustainablevillage.com
56 http://www.sourceopen.org/wiki/?pagename=OpenSourceEcology.About
57 http://www.sourceopen.org/wiki/?pagename=OpenSourceEcology.OverviewOfOsbornFacility
58 http://www.sourceopen.org/wiki/?pagename=OpenSourceEcology.Hydroponics
59 http://sourceopen.org/OSE_Future_Work_2007.pdf
60 http://www.eee-info.net
61 Beispiele bieten schon jetzt die Web-Auftritte von Bagni di Lucca in Italien (http://www.globalvillage-it.com/villaggio.php?id=progetto&rid=villaggio) und der des *Solar Power Village* in Tamera, Portugal (http://www.igf-online.org/?id=25).

5.5 Die Wiederentdeckung der Wissens – Empires

Einmal in Gang gesetzt, wird der Prozess der globalen Dörfer vielleicht die Welt überraschen. Eine ebenso große Überraschung könnte aber auch darin bestehen, dass das Prinzip vielleicht viel älter ist als bis dato angenommen. Die Hypothesen, dass manche Kulturen in früherer Zeit auf gemeinsam genutzten Wissen beruhten, dass es sehr phantasievolle nicht-digitale Wege gab, dieses Wissen zu kodieren und zu transportieren, dass dadurch in kürzester Zeit Reiche entstanden sind wie kein militärischer Eroberer sie jemals hätte vereinen können, sind in jüngster Zeit durch Forschungsergebnisse über das Reich der Inka unterstützt worden. Uwe Plachetka (2006) liefert eine faszinierende Zusammenschau von Biodiversität, landwirtschaftlicher Verwissenschaftlichung und Kommunikation im Inkareich, die zeigt, wie auf Basis von geteiltem Wissen auch in spektakulärem Ausmaß reale Macht und Handlungsfähigkeit entstand, die sich freilich nur als quasistaatliche Organisation behaupten konnte und historisch das Pech hatte, in der Konfrontation mit den spanischen Conquistadores auf einen überlegenen Weltherrschaftsanspruch zu treffen. Vielleicht lässt sich auf diese Art und Weise demonstrieren und vorhersehen, dass die Epoche der Monopolisierung des Wissens, deren Hochblüte und deren Ende wir gegenwärtig in paradoxer Parallelität erleben, eine viel kürzere Zeitdauer in der Geschichte hatte als derzeit angenommen und dass die gemeinsamen Interessen all derer, die ihr Leben und ihre Entwicklung auf geteiltem Wissen aufbauen, früher oder später auch zur materiellen Gewalt werden können und müssen.

Literatur

Ax, C. (2002), *Das Handwerk der Zukunft. Leitbilder für nachhaltiges Wirtschaften*, Birkhäuser Verlag.

Benkler, Y. (2006), *The Wealth of Networks: How Social Production Transforms Markets and Freedom*, Yale University Press, New Haven und London. http://www.benkler.org/wealth_of_networks/index.php/Main_Page [26. Jan 2007].

Fleissner, P. (2006), Die Heilung der Achillesferse, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 409–426. http://www.opensourcejahrbuch.de/download/jb2006/chapter_09/osjb2006-09-01-fleissner [26. Jan 2007].

Gershenfeld, N. (2005), *Fab: The coming Revolution on your Desktop – From Personal Computers to Personal Fabrication*, Basic Books, New York.

Groenemeyer, M. (2004), Wenn uns die Arbeit ausgeht . . . , *in* A. Exner, J. Sauer, P. Lichtblau, N. Hangel, V. Schweiger und S. Schneider (Hrsg.), 'Losarbeiten – Arbeitslos?', Unrast-Verlag, Münster. In Kooperation mit Attac.

Hänggi, M. (2006), 'USA: „Biolinux gegen Syngentasaurus Rex"', Gentech-news. http://www.blauen-institut.ch/Pg/pM/pM6/pm1445.html [25. Jan 2007].

Kelly, K. (1995), *Out of Control: The New Biology of Machines, Social Systems and the Economic World*, Perseus Books Group. http://www.kk.org/outofcontrol/contents.php [13. Feb 2007].

Kurz, R. (1999), 'Der Schwelbrand des globalen Finanzsystems: Das Elend des IMF und die pazifische Geldschöpfungsmaschine', *trend* (1). http://www.trend.infopartisan.net/trd0199/t240199.html [28. Jan 2007].

Kurz, R. (2003), 'Dollarinflation', *Neues Deutschland* **Nov 2003**. http://www.exit-online.org/link.php?tabelle=schwerpunkte&posnr=16 [27. Jan 2007].

Lyle, J. T. (1996), *Regenerative Design for Sustainable Development*, John Wiley & Sons, New York.

Margulis, L. und Case, E. (2006), 'The Germs of Life. Our Ancestors were Bacterial Communities', *Orion* (6). http://www.orionmagazine.org/pages/om/06-6om/Health.html [27. Jan 2007].

McLuhan, H. M. (1962), *The Gutenberg Galaxy*, University of Toronto Press, Toronto.

Merten, S. und Meretz, S. (2005), Freie Software und Freie Gesellschaft, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 293–309. http://www.opensourcejahrbuch.de/download/jb2005/chapter_05/osjb2005-05-03-mertenmeretz [12. Dez 2006].

Moravec, H. (1999), 'Rise of the Robots', *Scientific American* **281**(6), S. 124–135. http://www.frc.ri.cmu.edu/~hpm/project.archive/robot.papers/1999/SciAm.scan.html [27. Jan 2007].

Plachetka, U. (2006), 'Wissensbasierte Weltsysteme und biodiverse Landwirtschaft in den Anden', DorfWiki. http://www.dorfwiki.org/wiki.cgi?UweChristianPlachetka/WissensbasierteWeltsystemeUndBiodiverseLandwirtschaftInDenAnden [9. Feb 2007].

Reichwald, R. und Piller, F. T. (2006), *Interaktive Wertschöpfung. Open Innovation, Individualisierung und neue Formen der Arbeitsteilung*, Gabler. Die zentralen Teile des Buches stehen online unter http://www.open-innovation.com/iws/Reichwald-Piller_IWS-2006_Auszug_CC.pdf [27. Jan 2007] zur Verfügung.

Silvi, G. (1999), The Era of Generative Tech-Nets: Bit Watt -.- and Complexity, *in* 'Generative Art 1999', Mailand. http://www.piazzetelematiche.it/1994-2005/doc_e_genart.html [27. Jan 2007].

Todd, N. J. und Todd, J. (1994), *From Eco-Cities to Living Machines: Principles of Ecological Design*, North Atlantic Books, Berkeley, CA, USA.

Toffler, A. und Toffler, H. (2006), *Revolutionary Wealth*, Knopf, New York.

Wolfram, S. (2002), *A New Kind of Science*, Wolfram Media, Champaign, IL.

Zuboff, S. und Maxmin, J. (2004), *The Support Economy: Why Corporations Are Failing Individuals and the Next Episode of Capitalism*, Penguin, London.

Kapitel 2

Open-Source-Ökonomie

„Wirtschaftswissenschaft ist das einzige Fach, in dem jedes Jahr auf dieselben Fragen andere Antworten richtig sind."

– Danny Kaye

Open-Source-Ökonomie: Einleitung

DANIEL AUENER

(CC-Lizenz siehe Seite 563)

> „Es ist der Anfang [. . .] eines neuen Zeitalters, des Informationszeitalters." (Manuel Castells)

„Open Source fasziniert" heißt der erste Teil des vorliegenden Jahrbuchs, in dem sich das folgende ökonomische Kapitel befindet. Warum gerade die Ökonomie diesen Platz erhalten hat, mag mancher Leser fragen, denn Open Source fasziniert nicht nur Ökonomen, sondern auch Soziologen, Ethnologen, Rechtswissenschaftler und Informatiker. Wir sehen aber eine besondere Bindung von Open Source und Ökonomie, die in den letzten Jahren entstanden ist. Um diese Bindung zu verstehen, lohnt es sich, zunächst den Umschwung in der ökonomischen Lehre zu betrachten, der sich im beginnenden Informationszeitalter ereignet hat und sich am stärksten in der Einschränkung der Theorie des *homo oeconomicus* ausdrückt. Diese grundlegende Annahme der klassischen Ökonomie, der Mensch würde sich immer rein rational verhalten, lassen führende Ökonomen nicht mehr uneingeschränkt gelten: „Die Prognosen [dieses] ökonomischen Verhaltensmodells sind empirisch überprüft worden, und in vielen Fällen hat sich systematisch abweichendes Verhalten gezeigt", stellt der Schweizer Ökonom Bruno S. Frey treffend fest.

Inwiefern das aufkommende Informationszeitalter diesen Wandel in der Ökonomie vielleicht sogar verursacht hat, lässt sich an dieser Stelle nicht klären. Wir begnügen uns mit der These, dass die „neue Ökonomie" schon durch ihr zeitliches Auftreten als „Kind" des Informationszeitalters verstanden werden kann. Sie ist aber kein Einzelkind: Auch die Freie-Software- und Open-Source-Bewegung wurde im Informationszeitalter geboren und fand dort die richtigen Bedingungen, um sich entwickeln zu können.

Open Source und „neue Ökonomie" sind also vom gleichen Zeitgeist geprägt. Das ist aber nicht die einzige Bindung zwischen beiden. In den drei vorherigen Ausgaben des Open Source Jahrbuchs kann man sich von den bestehenden wechselseitigen Beziehungen ein gutes Bild machen: In den verschiedenen ökonomischen Artikeln untersuchten anerkannte Ökonomen unter anderem die unternehmerischen Möglichkei-

ten rund um Open-Source-Software, die Motivation von Open-Source-Programmierern oder das Innovationspotenzial des Open-Source-Entwicklungsmodells. Daraus lässt sich erkennen, dass die Open-Source-Praxis immer wieder moderne Theorien der Ökonomie bestätigt, welche wiederum der Open-Source-Bewegung helfen, sich selbst besser zu verstehen. Ein Beispiel dafür sind auch die beiden ersten Texte dieses Kapitels.

Der Artikel von Hal Varian und Carl Shapiro untersucht die „Ökonomie der Softwaremärkte". Diese, so stellen Varian und Shapiro fest, „lässt sich [. . .] eher vor dem Hintergrund aktueller Forschungsrichtungen [. . .] verstehen als auf der Grundlage der Basistheorien." So wie diese Arbeit der Open-Source-Gemeinde und dem geneigten Leser ein grundlegendes Verständnis der ökonomischen Umgebung von Open Source vermitteln kann, unterstützt Bruce Perens in seinem Artikel „Open Source – ein aufstrebendes ökonomisches Modell" die neuen ökonomischen Ansätze, indem er sie mit der praktischen Erfahrung eines Open-Source-Softwareentwicklers verknüpft. Dabei stellt er fest, dass „Open Source viel enger mit dem Phänomen des Kapitalismus verbunden ist, als man vielleicht erwartet hätte".

Der dritte Artikel dieses Kapitels stammt von Francis Heylighen und stellt das Prinzip der „stigmergischen Organisation" vor. Dieses kommt ursprünglich aus dem Tierreich und beschreibt das Prinzip, nach dem Termiten ihre Arbeit koordinieren. Angewendet auf Open Source erklärt es, wie die Arbeiten daran auch ohne die koordinierende Funktion von Markt oder Firma organisiert werden.

Abschließend lässt sich festhalten: Die beiden „Kinder" des Informationszeitalters Open Source und „neue Ökonomie" wachsen und werden erwachsen im gemeinsamen Zusammenspiel. Dabei ist es nicht nur so, dass Open Source eine besondere Faszination auf die Ökonomen ausübt. Andersherum erlauben es die neuen ökonomischen Ansätze auch zu erklären, was so faszinierend ist an Open Source.

Die Ökonomie der Softwaremärkte*

HAL VARIAN UND CARL SHAPIRO

(CC-Lizenz siehe Seite 563)

Im Vergleich zu anderen Produktmärkten ist der IT-Markt, insbesondere der Softwaremarkt, durch eine Reihe von Besonderheiten gekennzeichnet, die ihn von anderen Märkten unterscheiden. Dazu gehören die Komplementarität von Teilsystemen, hohe Transaktionskosten aufgrund von Netzwerkeffekten und langfristiger Kundenbindung (*lock-in*), lückenhafte Verträge im Bereich des Kundendienstes sowie das Abwägen zwischen Flexibilität und Kostenersparnis. Diese Besonderheiten haben bei der Wahl eines Produktes vor dem Hintergrund eines höchst dynamischen Marktes großen Einfluss auf das Verhalten von Konsumenten und Produzenten. Wir halten es daher für unerlässlich, offene Standards einzuführen, die sowohl den Wettbewerb als auch die Wahlmöglichkeiten am Markt optimieren.

Schlüsselwörter: Netzwerkökonomie · Switching-Kosten · offene Standards

1 Einleitung

Die Tatsache, dass der Markt für IT-Plattformen durch Netzwerkeffekte erheblich beeinflusst wird und in einer engen Wechselbeziehung zum Markt für IT-Anwendungen steht, macht die Softwareindustrie zu einer höchst komplexen Branche. Viele der ökonomischen Wechselbeziehungen, die für die Entwicklung der Branche bedeutend sind, lassen sich daher eher vor dem Hintergrund aktueller Forschungsrichtungen der Ökonomie verstehen als auf der Grundlage der Basistheorien, die im ersten Studienjahr gelehrt werden. Als weiterführende Lektüre empfehlen wir dazu unser Buch „Information Rules“ (Shapiro und Varian 1999).

Folgende Themenbereiche sind besonders relevant: Komplementarität zwischen den einzelnen Komponenten eines Informationssystems, Switching-Kosten als Mittel zur Konsumentenbindung und Umsatzsicherung, Zusicherungen des Verkäufers

* Aus dem Englischen von Christine Rolles.

Ökonom. Begriff	Definition	Implikation
Komplementariät	Der Wert eines Betriebssystems ist abhängig von der Verfügbarkeit kompatibler Anwendungen.	Vor der Entscheidung für ein System muss der genaue Bedarf ermittelt werden.
Switching-Kosten	Der Austausch einer einzelnen Komponente eines Computersystems kann sehr kostenintensiv sein.	Es sollten Entscheidungen getroffen werden, die künftige Flexibilität wahren.
Zusicherung des Verkäufers	Hersteller können Flexibilität oder niedrige Preise für die Zukunft versprechen, müssen sich aber nicht daran halten.	Es sollte auf grundlegende Zusicherungen geachtet werden, wie z. B. die Selbstverpflichtung zu offenen Schnittstellen.
Netzwerkeffekte	Der Wert einer Anwendungssoftware oder eines Betriebssystems kann stark davon abhängen, wie groß der Benutzerkreis ist.	Für ein geschlossenes Benutzernetzwerk ist die Standardisierung innerhalb des Netzwerkes wichtiger als die Wahl eines Industriestandards.
Lizenzvereinbarungen	Der dauerhafte Lizenzerwerb erfordert eine einmalige Zahlung, das Abonnement jährliche Beitragszahlungen.	Lizenzen können sich als besonders störend erweisen, wenn die Switching-Kosten hoch sind.
Bündelung	Anbieter möchten ihre Software gebündelt verkaufen, um so anderen den späteren Zugang zum Markt zu erschweren.	Der Kauf eines Softwarepakets kann zwar kostengünstiger sein, schränkt aber gleichzeitig die künftige Flexibilität und Wahlmöglichkeiten ein.

Tabelle 1: Zusammenfassung der ökonomischen Effekte und ihrer Auswirkungen

für die Zukunft als Strategie in Verhandlungen, Grunddefinition von Netzwerkeffekten, Nutzung von Lizenzbestimmungen zur Förderung unterschiedlicher Geschäftsmodelle sowie Software-Bündelung als Geschäftspraktik zur Umsatzsteigerung und Veränderung der Marktstruktur. Die Offenlegung des Quelltextes kann viele dieser Punkte beeinflussen. Eine erschöpfende Behandlung aller Aspekte der Netzwerkökonomie würde jedoch den Rahmen dieser Abhandlung sprengen; in Tabelle 1 sind die wichtigsten ökonomischen Begrifflichkeiten zu Softwaremärkten zusammengefasst.

Bei der Entwicklung von Richtlinien ist zu berücksichtigen, dass die Entscheidung für den freien Zugang zum Quelltext nur Teil einer größeren Debatte in der Computer- und Softwareindustrie ist. Innerhalb der Branche wird der Begriff *open computing* verwendet, um einen Ansatz zu beschreiben, der gleichfalls auf Hardware und Software zutrifft und Modularität, Interoperabilität, Interkonnektivität und Systemflexibilität hervorhebt. Der Schlüssel zu offenen Computersystemen ist in offenen Standards zu finden. Bekannte Open-Source-Projekte, wie z. B. Linux, erfüllen alle diese Anforderungen. Daher eignen sich Systeme, die auf der Grundlage von Linux entstanden sind, viel besser für das *open computing* als Systeme im Bereich von Plattformen, deren *APIs* urheberrechtlich geschützt sind. Etliche Vorteile der Open–Source-Software können sogar noch viel effektiver genutzt werden, wenn der Zugang zu ganzen Computersystemen offen ist.

Das trifft insbesondere auf die ökonomischen Aspekte von Offenheit zu. Da Hardware und Software, Server und Desktop-Computer, Plattformen und Anwendungen alles Komponenten eines einzigen Bereiches sind, kann der ökonomische Netzwerkeffekt Auswirkungen auf die gesamte Computer- und Softwarewelt haben. Daher kann Offenheit eine einschneidende Wirkung auf die industrielle Entwicklung haben. Offene Standards und Interoperabilität führen dazu, den Konkurrenzschwerpunkt in der Computerindustrie vom Wettbewerb um den Standard auf den Wettbewerb innerhalb des Standards zu verlagern. Weniger erfahrene Branchen benötigen wahrscheinlich Zeit, um verschiedene Ansätze auszuprobieren, bevor sie sich für einen Standard entscheiden. Wenn sich jedoch Konkurrenten mit anfangs verschiedenen Ansätzen erst einmal angenähert haben, existiert bereits ein Standard, gleichgültig, ob dieser „offiziell" anerkannt ist oder nicht. Wenn das Urheberrecht für einen Standard in der Hand eines einzigen Unternehmens liegt, bleibt wenig Spielraum für konkurrierende Unternehmen. Ist er jedoch für alle Branchenvertreter frei zugänglich, bleibt der Konkurrenzkampf hart. Verbraucher und Unternehmer profitieren davon, und die Anerkennung kommt aktuellen Innovatoren zugute und nicht jenen, deren Innovationen auf einer früheren Stufe industrieller Entwicklung erfolgreich waren.

2 Grundlagen ökonomischer Konzepte im Softwarebereich

Die Wahl einer bestimmten Hard- oder Softwareplattform erfordert eine genaue Betrachtung des gesamten Informationssystems. Bei der Entscheidung für ein System spielen Hardware, Software, Personal, Schulung, Systemadministration sowie weitere Faktoren eine wichtige Rolle; die isolierte Betrachtung einer einzigen Komponente kann daher ein falsches Bild vermitteln. Folglich ist die Wahl der Software schwieriger als viele andere Kaufentscheidungen. Sie hat nicht nur Auswirkungen auf Hardware, Schulung und Personal, sondern impliziert auch konzeptionelle Überlegungen bezüglich der Kosten für die gesamte Dauer der Nutzung und der Netzwerkökonomie. Verschiedene Softwareentwicklungsmodelle und unterschiedliche Zugangsstufen zum Quelltext können auf eine Reihe notwendiger Kalkulationen Einfluss haben, die bei einer gut durchdachten Entscheidung für eine Software empfehlenswert sind.

2.1 Switching-Kosten und lock-in

Das eben beschriebene Konzept der *Komplementarität* impliziert, dass die Komponenten eines Informationssystems sich in Abhängigkeit voneinander ergänzen. Jede Entscheidung, eine einzelne Komponente auszutauschen, wird vermutlich dazu führen, dass auch andere Komponenten ausgetauscht werden müssen. Neue Hardware könnte auch neue Anwendungsprogramme erforderlich machen. Ein neues Betriebssystem könnte der Anlass für die Wahl einer neuen Anwendungssoftware sein. Neue Anwendungsprogramme erfordern wiederum Fortbildung, und eine neue Serversoftware könnte eine Aktualisierung der Anwendungssoftware am Arbeitsplatz notwen-

dig machen. Diese Flut von Veränderungen bringt zwangsläufig Switching-Kosten mit sich. Sind diese Austauschkosten hoch, binden sie die Benutzer an ihr bisheriges Computersystem (*lock-in*) oder zumindest an einige seiner Komponenten.

In der Computerbranche sind hohe Austauschkosten und das *lock-in* von Anwendern keine Seltenheit. In vielen Fällen übersteigen die Gesamtaustauschkosten für das Unternehmen bei weitem die Anschaffungskosten für Hardware und Software. Sind die Kosten für eine Alternative hoch und ist der Nutzer bei der Zulieferung erforderlicher Komponenten eines Systems auf einen einzigen Anbieter angewiesen, unterliegt er unweigerlich der Willkür dieses Anbieters, d. h. er ist auch von dessen Service, Unterstützung, Lizenzverfahren und Preispolitik abhängig.

Viele Informationstechnologieanbieter vertrauen auf Switching-Kosten als wichtigem Teil ihrer Geschäftsstrategie. Wenn ein Benutzer sich erst einmal für einen bestimmten Datenbankhersteller oder ein Betriebssystem entschieden hat, könnte ihn ein Wechsel teuer zu stehen kommen. Die Austauschkosten liefern ihn dem Händler auf Gedeih und Verderb aus. Clevere Kunden sollten also nicht nur direkt auf das Geschäft als solches schauen, sondern auch auf die gesamte Lebensdauer des Produktes. Wenn die Kosten für eine Alternative künftig sehr hoch ausfallen, bleibt dem so gebundenen Kunden wenig Spielraum für Verhandlungen. Konsumenten sollten immer mit Preiserhöhungen für spätere informationstechnologische Dienstleistungen rechnen, die in ihrem Erstkaufvertrag nicht enthalten sind.

Entscheidungen für Informationssysteme erfordern vom Konsumenten also *vorausschauende* Planung und *rückblickende* Erkenntnis. Wenn er sich lediglich auf die aktuelle Situation konzentriert, kann dies zu Fehlentscheidungen führen. Informationssysteme sind langfristige Investitionen, die Konsumenten für lange Zeit an bestimmte Anbieter binden können. Von Erstentscheidungen, welche die spätere Flexibilität optimieren, profitieren Konsumenten am meisten.

2.2 Zusicherungen des Verkäufers

Im Falle von hohen Austauschkosten ist das Verhältnis zwischen Käufer und Anbieter grundsätzlich angespannt: Der Käufer möchte Flexibilität bewahren, während der Verkäufer das *lock-in* unterstützt. Anbietern ist durchaus bewusst, dass Käufer sich nur ungern an urheberrechtlich geschützte Produkte binden und versuchen daher, den Umfang der Bindung herunterzuspielen.

Open-Source-Produkte verändern die Dynamik dieser Verhandlungen. Sie bieten Verkäufern die Möglichkeit, sich ihren Kunden gegenüber engagierter zu zeigen, nachdem diese sich für ein Informationssystem entschieden haben. Wenn der Quelltext einer Software frei verfügbar ist, können die Nutzer, ggf. mit der Unterstützung von Dritten, ihre Softwareinvestitionen flexibler gestalten, d. h. beibehalten oder erweitern. Diese Option ermöglicht Anwendern, Open-Source-Lösungen zu wählen, da sie bis zu einem gewissen Grad sicher sein können, dass die Austauschkosten relativ niedrig bleiben. Sollten sie dann mit ihren aktuellen Anbietern unzufrieden

sein, können sie zu anderen wechseln, wobei ihre eigenen Switching-Kosten noch überschaubar bleiben. Kurz gesagt, niedrige Austauschkosten fördern den Wettbewerb und zwingen Anbieter, sich kundenorientiert zu verhalten und auch nach dem Erstverkauf entsprechend gute Konditionen zu gewähren. Kundenorientierte Softwareanbieter hinterlegen seit langem den Quelltext ihrer Produkte bei Treuhändern – das garantiert den Kunden, im Falle eines Firmenkonkurses nicht mit leeren Händen dazustehen.Open-Source-Produkte bieten jedoch noch viel mehr Sicherheit: Sie begrenzen das Ausmaß opportunistischen Verhaltens und schaffen zudem ein wettbewerbsorientierteres Umfeld für Verkäufer.

Dieser Effekt entsteht jedoch nur auf der Grundlage einer weiteren Öffnung, die über den frei zugänglichen Quelltext hinausgeht. Ein Teil der Branche verwendet den Begriff *open computing*, um einen Ansatz zu beschreiben, der sowohl für Hard- als auch für Software Modularität, Interoperabilität, Interkonnektivität und Systemflexibilität hervorhebt. Der Schlüssel zur frei zugänglichen Computeranwendung liegt bei offenen Standards, die Schnittstellen im Bereich der Hardware und *APIs* im Bereich der Software. Bekannte Open-Source-Produkte, wie z. B. Linux, bieten alle diese Vorteile; Systeme, die auf Linux aufbauen, sind daher als völlig offene Computersysteme viel leichter zu warten als Systeme, die auf Plattformen mit lizenzgebundenen Schnittstellen in der Hand einiger weniger Anbieter aufbauen. Viele der Vorzüge, die der Open-Source-Software zugeschrieben werden, können durch die vollständige Öffnung von Computersystemen noch optimiert werden.

2.3 Netzwerkeffekte und positives Feedback

Wenn der Wert eines Produktes oder einer Dienstleistung davon abhängt, wie viele andere Nutzer es dafür gibt, sprechen Ökonomen von einem *Netzwerkeffekt*. So hängt z. B. der Wert eines Faxgerätes davon ab, wie viele weitere Faxgeräte vorhanden sind. Ebenso kann der Wert eines E-Mail-Benutzerkontos davon abhängen, wie viele der Menschen, mit denen man Korrespondenz pflegt, E-Mail nutzen.

In manchen Fällen wird der Wert eines Produktes durch die Verbreitung eines anderen Produktes bestimmt. Ein DVD-Player gewinnt z. B. dann an Wert, wenn die Anzahl der verfügbaren DVDs steigt. Netzwerkeffekte, die über solche komplementären Produkte entstehen, werden als *indirekte Netzwerkeffekte* bezeichnet.

Bei der Computersoftware sind die indirekten Netzwerkeffekte sehr ausgeprägt, da der Wert eines Betriebssystems bis zu einem gewissen Grad davon abhängt, wie viele Anwendungsprogramme darauf laufen. Gleichzeitig steigt der Wert einer Anwendung, wenn sie auf einem weit verbreiteten Betriebssystem läuft. Solche indirekten Netzwerkeffekte sind womöglich für einen Server weniger von Bedeutung; meist kommt es nur darauf an, dass eine bestimmte Anwendung, z. B. ein Webprovider oder eine Datenbank, auf diesem Server laufen. Die meisten Nutzer interessiert es ebenso wenig, mit welchem Betriebssystem ihre Registrierkasse arbeitet. Jedoch gibt es auch Fälle, in denen der Nutzer beim Kauf eines Betriebssystems noch nicht weiß, mit welchen

Programmen sie oder er arbeiten will. In diesen Fällen ist das System, auf dem die meisten Anwendungen laufen, wohl das attraktivste, weil es künftige Optionen offen lässt: Standardanwendungsprogramme sind verfügbar, der Austausch von Dateien ist einfach, Angestellte, Kunden, Partner oder Freunde sind höchstwahrscheinlich vertraut mit dem System usw. Die unmittelbare Attraktivität, die diese Systeme nur aufgrund ihrer weiten Verbreitung besitzen, bedeutet auch, dass die marktführenden Betriebssystem- und Anwendungsanbieter den alternativen Anbietern gegenüber im Vorteil sind, selbst wenn die Alternative qualitativ gleichwertig ist.

2.4 Die Bündelung von Software

Software wird häufig gebündelt, d. h. als Paket verkauft. *Microsoft Windows* selbst besteht aus einer großen Zahl von unterschiedlichen Programmen, die zusammenarbeiten; *Microsoft Office* beinhaltet mehrere, unterschiedliche *productivity tools*, also leistungssteigernde Programme, die interoperationell sind. *Red Hat*, ein Linux-Standardprodukt, beinhaltet Hunderte von interoperationellen Programmen.

Die Bündelung von Software ist daher sowohl für Verkäufer als auch für Käufer attraktiv, obwohl ein spezielles Softwarepaket womöglich nur den Interessen einer der beiden Parteien gerecht wird. Käufer werden es wahrscheinlich begrüßen, dass sie ein komplettes, integriertes Softwarepaket mit Anwendungen erhalten, die untereinander kompatibel sind und die ihren künftigen Bedarf an Software komplett abdecken. Verkäufer bieten diese Pakete vermutlich an, weil sie damit den Bedürfnissen ihrer Kunden besser gerecht werden und womöglich auch noch den Verkauf um weitere Softwarekategorien erweitern können. Vielleicht entscheidet sich jemand zunächst für *Microsoft Office*, weil sie oder er mit *Microsoft Word* arbeiten möchte. Wenn diese Person dann später eine Tabellenkalkulation benötigt, dann wird sie sich selbstverständlich für *Microsoft Excel* entscheiden, da sie das Programm als Teil des Paketes bereits erworben hat und sich höchstwahrscheinlich nicht nach einem Konkurrenzprodukt umsehen oder es gar kaufen wird. Wenn diese Effekte vorherrschen, dann kann es für Anbieter individueller Einzelsoftware, wie z. B. Tabellenkalkulationen, sehr schwer werden, sich auf dem Markt zu behaupten. Wenn mit dem Austausch einzelner Komponenten Kosten verbunden sind, müssen die Gesamtaustauschkosten für das Paket aus der Summe der Einzelkosten ermittelt werden. Auch wenn jede einzelne Komponente vertretbare Switching-Kosten beinhaltet, können die Gesamtkosten erheblich sein und zu einem *lock-in* führen. Darüber hinaus kann die Bündelung von Software dazu führen, dass Anwender Migrationsstrategien, d. h. einen Wechsel der Plattform, meiden, wenn nur eine Komponente auszutauschen ist.

Literatur

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

Open Source – ein aufstrebendes ökonomisches Modell*

BRUCE PERENS

(CC-Lizenz siehe Seite 563)

Die Entwickler von Open-Source-Software haben, vermutlich ohne bewusste Absicht, ein neues und erstaunlich erfolgreiches Modell der Softwareentwicklung geschaffen. Durch eine genauere Untersuchung dieses Modells können eine Reihe wichtiger Fragen beantwortet werden. Da nicht jedem sofort ersichtlich ist, wie Open Source auch wirtschaftlich funktionieren kann, wird diese Form der Softwareentwicklung leider häufig als auf lange Sicht unwirtschaftlich verkannt. Mancher befürchtet gar, dass die potenziellen negativen Auswirkungen auf die proprietäre Softwareindustrie der Gesamtwirtschaft Schaden zufügen können. Bei einer genaueren Betrachtung der allgemeinen wirtschaftlichen Funktionsweisen des Softwaresektors ist es jedoch ein Leichtes, zu zeigen, dass Open Source sowohl zukunftsfähig als auch von enormem Vorteil für die Gesamtwirtschaft ist. Konventionelle Theorien der freien Marktwirtschaft lassen sich eins zu eins auch auf Open Source anwenden. Dabei stellt sich heraus, dass Open Source viel enger mit dem Phänomen des Kapitalismus verbunden ist, als man vielleicht erwartet hätte.

Schlüsselwörter: Wirtschaftlichkeit · Produktdifferenzierung · ökonomische Modelle der Softwareentwicklung

1 Eine solide wirtschaftliche Grundlage

In den Anfängen der Open-Source-Bewegung haben auch Befürworter den wirtschaftlichen Aspekt des Modells nicht vollständig verstanden. Aufgrund dieses Mangels an Verständnis erzeugten wir den Eindruck, dass Open Source[1] keine solide wirtschaftliche Grundlage habe und trugen so zu der allgemeinen Ansicht bei, dass Open

* Aus dem Englischen von Katharina Mertens, Nadja Schüler, Angela Dirlick und Martin Görnitz.

1 Im Rahmen dieses Artikels können *Open Source* und *freie Software* als gleichbedeutend behandelt werden. Es existieren philosophische Differenzen zwischen den Vertretern beider Bewegungen, aber auf einen

Source weder zukunftsfähig sei, noch in der Lage, das Bedürfnis des Marktes nach neuer Technologie zu befriedigen. Dieser Eindruck bedarf inzwischen dringender Korrektur. Eric Raymond (1999) versuchte in seinem Werk „The Cathedral and the Bazaar“[2], Open Source als eine Art Geschenkökonomie zu erklären, ein Phänomen unter Computerprogrammierern, die Spaß daran haben, sich auf kreative Art und Weise einmal mit etwas zu beschäftigen, das nichts mit ihrer beruflichen Anstellung zu tun hat, und sich aus einem gestalterischen Anspruch heraus wünschen, dass ihre Arbeit wertgeschätzt wird. Raymond beschreibt dabei äußerst zutreffend das Verhalten von Programmierern in ihrer eigenen Subkultur: Die von ihm beschriebenen Beweggründe beherrschten tatsächlich zu Beginn die Open-Source-Szene und motivieren auch heute noch eine ganz entscheidende Gruppe der Mitwirkenden im Bereich Open Source.

Raymond suchte jedoch nicht nach Erklärungen dafür, warum sich große Unternehmen wie *IBM* an Open Source beteiligen, dazu kam es erst, nachdem sein Werk bereits erschienen war. Zum betreffenden Zeitpunkt hatte die Geschäftswelt gerade erst begonnen, sich ernsthaft für Open Source zu interessieren, und es handelte sich noch um kein bedeutsames wirtschaftliches Phänomen. „The Cathedral and the Bazaar“ kann daher nicht auf die Art von Einblicken in die wirtschaftliche Funktionsweise von Open Source zurückgreifen, die uns heute zur Verfügung stehen.

Diese ersten Ausführungen von Raymond wurden leider häufig fälschlicherweise als Beweis für das schwache ökonomische Fundament von Open Source interpretiert. Raymond zufolge wird der eigene Arbeitsaufwand eher mit immateriellen als mit finanziellen Gegenleistungen honoriert. Heute jedoch lässt sich erfreulicherweise feststellen, dass es für viele Entwickler von Open-Source-Software eben doch sehr hohe Verdienstmöglichkeiten gibt. Diese sind jedoch noch immer indirekter als in der Entwicklung proprietärer Software. Um die Ökonomie von Open Source zu verstehen, muss man daher tiefer vordringen, als dies bei der Betrachtung proprietärer Software notwendig wäre.

Großteil der Software, die per Definition Open Source ist, kann auch die Definition freier Software der *Free Software Foundation (FSF)* angewandt werden.

Es ist wichtig zu wissen, dass zu dem Zeitpunkt, an dem ich das Dokument, welches später als *Open Source Definition* bekannt wurde, als Richtlinie für das Debian-Projekt entwarf, die *FSF* ihre Definition freier Software noch nicht veröffentlicht hatte. Damals kommentierte Richard Stallman in einer persönlichen E-Mail, dass mein Dokument eine gute Definition freier Software darstellte.

Stallman widersprach jedoch meinem Argument, dass Unternehmen besser beraten wären, auf eine Free-Software-Lizenz zu verzichten, wenn dadurch der Umfang der Differenzierung für dieses Unternehmen geschmälert würde. Aus diesem Grund verwende ich im Folgenden die Bezeichnung Open Source, um Stallmans Ansichten bezüglich freier Software zu würdigen.

2 Raymond ergänzte sein Essay um zwei weitere Werke, „Homesteading the Noosphere“ (1998) und „The Magic Cauldron“ (1999), in denen er seine Betrachtung der Open-Source-Ökonomie fortsetzt. Alle drei Texte können unter http://www.catb.org/~esr/writings/ abgerufen werden.

2 Wie wird Ihr Unternehmen so erfolgreich wie Microsoft?

Angehenden Unternehmern der Softwarebranche wird häufig die Frage gestellt, wie sie es schaffen wollen, ebenso erfolgreich zu sein wie *Microsoft*. Wer sich mit seinem Unternehmen gar auf Open Source spezialisiert, wird zudem häufig vor die Frage gestellt, wie man mit freier Software an den Erfolg von *Microsoft* anschließen kann. Doch diese Frage ist schon an sich unberechtigt, wenn es doch das eigentliche Ziel ist, die Dinge besser zu machen als *Microsoft*. Sie spiegelt die Tatsache wider, dass bis zum heutigen Zeitpunkt viele Leute Software von einer sehr auf den Anbieter ausgerichteten Perspektive aus betrachten.

Fast jeder – ob er es zugibt oder nicht – lässt sich von der Finanzkraft und Überheblichkeit beeindrucken, die mit *Microsoft* assoziiert wird, und wenn es um die Herstellung von Software geht, denkt jeder gleich an *Microsoft* und deren Vorgehensweise. Es stellt sich jedoch heraus, dass das Microsoft-Modell nur einen Bruchteil der Software, die in der heutigen Geschäftswelt hergestellt und benutzt wird, für sich verbuchen kann. Nur etwa 30 Prozent der entwickelten Software wird auch als Software verkauft.[3] Ein großer Teil steht überhaupt nicht zum Verkauf. Er wird direkt für den Kunden entwickelt, von den eigenen Mitarbeitern oder externen Dienstleistern, die ihre Entwicklertätigkeit und nicht das Endprodukt in Rechnung stellen. Man muss sich vor Augen führen, warum dies so ist, um die ökonomischen Aspekte von Open Source nachvollziehen zu können.

3 Open Source als Reaktion auf eine allgemeine Unzufriedenheit

Im Februar 1998 gründeten Eric Raymond und ich die Open-Source-Initiative. Wir konnten auf exzellente Vorarbeit aufbauen: Die Free-Software-Kampagne, die auf Richard Stallman zurückgeht, gab es bereits seit 1983[4] und durch sie wurde ein großer Teil dessen geschaffen, was man heutzutage als Open Source bezeichnet. Die Verbindung des Linux-Betriebssystemkernels mit Stallmans GNU-System wurde nach

3 In einem Bericht des *U.S. Office of Technology and Electronic Commerce* (2003) zur Größe des US-Softwaremarkts repräsentiert *Packaged Software* 24,6 Prozent der Softwareindustrie. Der Rest des Entwicklungssektors besteht aus *Computer Programming Services*, *Computer Integrated Systems Design*, *Computer Processing and Data Preparation and Processing*, *Information Retrieval Services*, *Computer Facilities Management Services* sowie verschiedenen Untergruppen von *Software Publishing*.

Allerdings beinhaltet der Bericht keine Angaben zu Software, die für den firmeninternen Gebrauch durch angestellte Programmierer in einem Unternehmen, dessen Hauptprodukte nicht softwarebezogen sind, hergestellt wird. Zurückhaltend geschätzt würde ich die Zahl der internen Entwickler im Bereich der externen Auftragnehmer ansiedeln, die unter *Computer Programming Services* mit 19 Prozent Marktanteil angegeben werden.

4 Das GNU-Projekt wurde ursprünglich im September 1983 angekündigt (siehe http://www.gnu.org/gnu/initial-announcement.html), obwohl das Projekt erst im Januar 1984 begann. 1985 schließlich kam es dann zur Gründung der *Free Software Foundation*. Ein Überblick des GNU-Projekts findet sich unter http://www.gnu.org/gnu/gnu-history.html.

und nach auch für den Einsatz in Unternehmen brauchbar. Jedoch stellte Stallman Open Source auf eine Weise dar, welche die Akzeptanz der Werte gewisser Freiheiten *a priori* voraussetzte. Stallman ist Programmierer, und er entschied sich daher für eine philosophische Darstellung, die Programmierer ansprach.

Geschäftsleute hingegen sind Pragmatiker und lassen sich von wirtschaftlichen Vorteilen beeindrucken. Da die Art seiner Darstellung Stallmans Zielgruppe von vorneherein begrenzte, hatte seine Kampagne nicht die wirtschaftliche Schlagkraft, die sich heute abzeichnet.[5] Raymond und ich entschieden uns dafür, Geschäftsleute auf eine pragmatischere Art und Weise anzusprechen, in der Annahme, dass sie Stallmans Theorie eher wertschätzen würden, wenn sie einmal deren konkrete Vorteile erkannt hätten.[6]

Der Begriff Open Source kam der Öffentlichkeit zum ersten Mal in einer Bekanntmachung zu Ohren, die ich auf *Slashdot* und einigen Mailinglisten veröffentlichte. Sie beinhaltete eine Einführung und die Open-Source-Definition, die sowohl ein Open-Source-Grundsatzprogramm als auch eine Definition einer akzeptablen Vorgehensweise zur Open-Source-Software-Lizenzierung darstellte. Diese hatte ich sechs Monate zuvor als Grundsatzprogramm für das Debian-Projekt ausgearbeitet. Eric Raymond überarbeitete seinen Artikel „The Cathedral and the Bazaar", der ein Jahr zuvor erschienen war, und ersetzte darin die Worte *Free Software* durch *Open Source*.

Einige Zeit später fragte ein Reporter Steve Ballmer, den damaligen Präsidenten von *Microsoft*, ob man plane, *Windows* als Open-Source-Produkt zur Verfügung zu stellen. Ballmer erklärte daraufhin, dass Open Source nicht nur ein bestimmter Quellcode sei, sondern eine bestimmte Art der Software-Lizenzierung – der Präsident von *Microsoft* hatte mein Grundsatzprogramm gelesen und wurde von der Presse dazu befragt!

Veröffentlichungen von so unbekannten Menschen, wie Raymond und ich selbst es damals waren, erfahren diese Art von Aufmerksamkeit gemeinhin nur dann, wenn die

5 Hinzu kam, dass das GNU-Projekt eine Menge Software entwickelt und integriert hatte, es jedoch nicht geschafft hatte, einen eigenen Betriebssystemkernel zu entwickeln. Dies öffnete die Tür für Linus Torvalds, den letzten Teil des Projekts zu vollenden. Üblicherweise bezieht ein Betriebssystem seinen Namen vom Kernel, ungeachtet der Tatsache, dass es viele andere Komponenten abseits des Kernels enthält. Hier wiederfuhr Stallman ein Unrecht, das bis heute fortbesteht. Obwohl Stallman einen Großteil der Arbeit vollbrachte, die *Linux* erst möglich machte, fühlte sich Torvalds Team von Kernelentwicklern ihm nicht verpflichtet. In den Ankündigungen von *Linux* fand die *FSF* keine Erwähnung. In Wahrheit ist *Linux* jedoch nur der Kernel. Der Rest dessen, was üblicherweise in einer Linux-Distribution enthalten ist, besteht größtenteils aus den Komponenten von Stallmans *GNU System*. Stallman sind daher sowohl das GNU/Linux-Konzept, die Entwicklung des C-Compilers (der sicher in seiner Bedeutung dem Linux-Kernel ebenbürtig ist), der Emacs-Editor als auch die Anregungen, die er zur Entwicklung eines Großteils des übrigen Systems gab, zuzuschreiben.

6 In gewissem Umfang würdigte Raymond dabei Stallmans Ansichten herab, als er einen Konflikt mit Stallman und der *FSF* hätte vermeiden können, wenn er sich, der Einfachheit halber, nur an die Gruppe gewandt hätte, für die Open Source von Interesse ist und die sich von der, die Stallman anspricht, grundsätzlich unterscheidet. Meiner Ansicht nach beruht das bedauernswerte Resultat eher auf den gegensätzlichen Persönlichkeiten als auf unvereinbaren Philosophien.

Welt aus irgendeinem Grund auf sie gewartet hat. Ich behaupte, dass der Grund eine allgemeine Unzufriedenheit mit dem Marktführer im Bereich Softwareentwicklung war, nämlich *Microsoft* und seinen Produkten, das gemeinhin als ein Unternehmen galt, welches die eigenen Interessen über die des Kunden stellte. Es war bekannt, dass ihre Software unbehobene *Bugs* enthielt und dass ihr Desktop-Betriebssystem extrem anfällig für Systemabstürze war. Sie kamen damit „durch", da es keine nennenswerten Konkurrenten im Desktopbereich gab.

Die Menschen suchten nach einer Alternative. Wenn wir Software an sich verbessern wollten, mussten wir anders vorgehen als *Microsoft*. Open Source schaffte neue ökonomische Beziehungen. Eine neue Peer-to-Peer-Beziehung entstand zwischen Geschäftskunden von Softwareprodukten, die jetzt in der Lage waren, direkt an der Entwicklung von Open-Source-Software mitzuwirken und somit zu Softwareentwicklern füreinander wurden. Softwareanbieter entwickelten ein neues Verhältnis zueinander, da sie gemeinsam an Open-Source-Projekten arbeiteten, während sie in anderen Bereichen Konkurrenten waren. Und auch das Verhältnis von Kunden und Anbietern veränderte sich, da nun beide gemeinsam an der Entwicklung von Software beteiligt sein konnten. Aufgrund der Tatsache, dass viele Hersteller auf denselben Quelltext Zugriff hatten und somit das Produkt unterstützen konnten, verlor der *lock-in*, die unfreiwillige Gebundenheit an einen Hersteller, an Bedeutung, und die Beziehung zwischen Kunde und Anbieter war von weniger Exklusivität geprägt. Als Ergebnis all dieser Veränderungen fanden sich neue Anbieter auf dem Markt, und auch die Etablierten ersetzten nach und nach proprietäre Software durch Open-Source–Produkte.

Die gemeinschaftliche Beteiligung an der Erschaffung, dem Eigentum und den Vorteilen der Software stand in starkem Gegensatz zu dem Modell von *Microsoft*. Erste Anzeichen einer überraschenden Effizienz von Open Source waren die Schaffung eines technisch erfolgreichen Betriebssystems und der ersten praktisch anwendbaren Webserver und -clients. An diesem Punkt begann die Öffentlichkeit, sich massiv für Open Source zu interessieren und darin zu investieren. Es folgte ein enormes wirtschaftliches Wachstum: Der Linux-Sektor allein hat eine jährliche Wachstumsrate von 37 bis 45 Prozent, und bis zum Jahr 2008 erwartet man einen Marktumfang von 35 Milliarden US-Dollar (IDC Software Consulting 2004).

Diese ungewöhnliche Akzeptanz und die verblüffenden finanziellen Aussichten sprechen dafür, dass Open Source Ansprüche bedient, die vorher nicht in gewünschter Weise befriedigt wurden. Wäre dem nicht so, hätten sich keine so erfolgreichen und scheinbar absurden Phänomene entwickelt:

Linux, einst das Hobby eines Studenten Anfang 20, erobert den Enterprise-Computing-Markt.

IBM, der Inbegriff eines konservativen Unternehmens, zieht seinem Milliarden Dollar teuren AIX-Betriebssystem ein Produkt vor, das von einer zusammengewür-

felten Gruppe von Programmierern ohne gemeinsames finanzielles Interesse entwickelt wurde, die dem Unternehmen nicht weisungsgebunden sind, mit einem Vorsitzenden, dessen gesamter Einfluss auf dem Respekt der anderen beruht.

Microsoft hat seit einem Jahrzehnt den ersten ernsthaften Konkurrenten: Programmierer, welche die Produkte ihrer Arbeit verschenken.

All das erscheint zunächst keinen Sinn zu ergeben und den allgemeinen ökonomischen Grundregeln für den Bereich der Technologieentwicklung zu widersprechen. Es handelt sich hier um ein neues wirtschaftliches Phänomen, das sich nur erklären lässt, wenn man die wirtschaftlichen Gegebenheiten der Softwareentwicklung näher betrachtet.

4 Die Ursachen des wirtschaftlichen Erfolgs

Da *Microsoft* nach Ansicht der meisten Menschen als das bedeutsamste Modellunternehmen im Bereich der Softwareerzeugung gilt, und da wir uns hier mit den wirtschaftlichen Gegebenheiten der Softwarebranche befassen, sollten wir uns eine Frage stellen: Was macht die wirtschaftliche Stärke von *Microsoft* aus?

Man könnte zunächst vermuten, es sei der Wohlstand von Bill Gates, aber das ist natürlich Unsinn. Sein Reichtum wird von vielen geneidet, aber Bill Gates ist nicht gleichzusetzen mit *Microsoft*. Und auch die Tatsache, dass die Firma aktuell über einen liquiden Bestand von 70 Milliarden Dollar verfügt, somit das größte Softwareunternehmen der Welt ist und 71 000 Mitarbeiter in 103 Ländern beschäftigt, beantwortet unsere Frage nicht. Die Bedeutung von *Microsoft* ist vielmehr darauf zurückzuführen, dass viele Unternehmen, nämlich seine Kunden, durch Microsofts Software effizienter arbeiten können und sogar ohne die entsprechende Software überhaupt nicht mehr agieren könnten. *Microsoft* stellt Werkzeuge her, und sein wirtschaftlicher Einfluss ist marginal, verglichen mit der gemeinsamen Bedeutung all derer, die diese Werkzeuge verwenden. Der sekundäre ökonomische Einfluss all der Menschen und Unternehmen, die eine Technologie nutzen, ist größer als der primäre ökonomische Einfluss des Geldes, welches durch den Verkauf der Technologie eingenommen wird. Und all das gilt natürlich auch für Open-Source-Software.

5 Die wirtschaftliche Bedeutung von Software in Unternehmen

Wir müssen uns vor Augen führen, dass die meisten Unternehmen nicht im Bereich der Softwareherstellung tätig sind: Sie bieten Dienstleistungen an, verkaufen Schiffe, Schuhe oder andere Produkte. Dennoch sind alle Unternehmen bis auf die allerkleinsten in einem gewissen Maße von Software abhängig. Ohne sie wäre ihre Arbeit weniger effizient oder sogar überhaupt nicht durchführbar. Erinnern wir

uns einmal daran, wie Finanzplanung ohne Tabellenkalkulationsprogramme aussah, Geschäftskorrespondenz ohne E-Mails und an Kundeninterfaces bestehend aus Telefon mit Tonwahlverfahren als fortschrittlichstem Instrument der Kundeninteraktion. Schnell wird klar, dass im heutigen täglichen Geschäftsleben Software unabdingbar ist. Sie wird so dringend benötigt, dass jedes Unternehmen mit mehr als 50 Mitarbeitern, auch wenn es keine Software verkauft, einen Programmierer, einen Webdesigner oder einen Skript-programmierenden Systemadministrator beschäftigt.[7] Diese Unternehmen stellen keine Software her, aber dennoch ist Software für sie im täglichen Geschäftsleben von größter Wichtigkeit.

6 Gebrauchstechnologie im Vergleich zu Produktdifferenzierung

Gebrauchstechnologie ist unverzichtbar für ein Unternehmen, aber nicht das Produkt, das verkauft wird. Wer Bücher verkauft, für den sind Bücher das Ertragszentrum, während Software ein Kostenzentrum ist, eine unvermeidbare Ausgabe im Geschäftsleben. Die polemische Aussage, IT spiele keine Rolle, bedeutet im Grunde genommen, dass IT häufig kein Ertragszentrum ist, aber dennoch insofern benötigt wird, als dass sie Geschäfte überhaupt erst ermöglicht. IT-Technologie ist Gebrauchstechnologie.

Es gibt zwei grundlegende Arten kostenproduzierender Gebrauchstechnologie: differenzierende und nicht differenzierende. Differenzierende Technologie macht ein Unternehmen im Vergleich zu einem Konkurrenten für den Kunden ansprechender. Wenn man zum Beispiel die Website Amazon.com besucht, um ein bestimmtes Buch zu kaufen, dann bietet einem *Amazon* auch andere Bücher an, die von Kunden erworben wurden, die sich für dieses Buch entschieden haben. Die vorgeschlagenen Bücher sind tatsächlich häufig so interessant, dass man eines davon zusätzlich erwirbt. Auf der Seite von *Barnes & Noble* gibt es diese Funktion nicht, und es ist daher nicht verwunderlich, dass *Amazon* mehr Bücher online verkauft. Die Software, die diese Empfehlungen möglich macht, unterscheidet *Amazon* also von anderen Anbietern.[8] Offensichtlich wäre es ein Fehler, eine solche differenzierende Software als Open Source zur Verfügung zu stellen, da sie in diesem Fall von einem Konkurrenzunternehmen genutzt werden könnte, um genauso ansprechend für die Kunden zu werden.

---

7 Das *U.S. Bureau of Labor Statistics* führt 2 249 000 Stellen in programmierbezogenen Berufen in den USA: 455 000 Computerprogrammierer, 800 000 Softwareentwickler, 994 000 Systemanalysten, Datenbankadministratoren und Informatiker. Nicht enthalten sind die 779 000 Computer-Support–Spezialisten und Systemadministratoren, die zumindest ab und an Skriptprogrammierung betreiben, sowie Computeranwender, die 149 000 Positionen besetzen (U.S. Bureau of Labor Statistics 2006).

8 Ironischerweise ist dieses Feature Anlass einer Patentrechtsklage, die *Cendant* im November 2005 gegen *Amazon* eingereicht hat. Nach einer Erneuerung der Klage durch *Cendant* im Juni 2005 und einer Gegenklage von *Amazon* ist „Cendant vs. Amazon" bis heute noch nicht entschieden.

Auf der anderen Seite würde es einen Unternehmen nicht schaden, wenn der Konkurrenz die Funktionsweise ihrer nicht-differenzierenden Software bis ins kleinste Detail bekannt wäre. Die Konkurrenz könnte sich sogar als ihr bestmöglicher Partner herausstellen, wenn sich die Zusammenarbeit auf nicht-differenzierende Software beschränkt, einfach weil die Bedürfnisse der Konkurrenz den eigenen so ähnlich sind. Dieses Prinzip wird Tag für Tag in der Welt der Open Source bestätigt. *HP* und *IBM* entwickeln gemeinsam Software, die beiden Anbietern dabei hilft, ihre Systeme zu verkaufen, während sie in anderen, spezielleren Bereichen der Softwareentwicklung, in denen eine Differenzierung möglich und sinnvoll ist, erbitterte Konkurrenten bleiben.

Bis zu 90 Prozent der Software eines jeden Unternehmens ist nicht-differenzierend. Ein großer Teil davon gehört zur Infrastruktur, der Grundlage, auf der differenzierende Technologie aufbaut. Unter Infrastruktur versteht man Dinge wie Betriebssysteme, Webserver, Datenbanken, Java-Application-Server und andere Middleware, Graphical-User-Interface-Desktops und die Tools, die allgemein bei GUI-Desktops benutzt werden, wie beispielsweise Webbrowser, E-Mail-Clients, Tabellenkalkulationsprogramme, Textverarbeitung und Präsentationsanwendungen. Jede Software, die für ein Nicht-Softwareunternehmen differenzierende Effekte hat, setzt auf einer oder mehrerer solcher Infrastrukturkomponenten auf. Ein wichtiger Indikator dafür, ob Software differenzierend ist oder nicht, ist die Frage, ob einem Konkurrenten dieselbe Software zur Verfügung steht. Weder Microsoft-Software, noch *Linux* und Open Source können auf lange Sicht dabei helfen, ein Unternehmen von anderen abzuheben, da sie jedem zur Verfügung stehen. Sie unterscheiden sich voneinander, können aber ihr Unternehmen nicht von anderen differenzieren. Durch die Wahl des einen oder anderen können Sie vielleicht Geld sparen oder die eigene Effizienz erhöhen, aber in der Regel nicht attraktiver für Ihre Kunden werden.

Ein weiterer wichtiger Indikator dafür, ob eine Software differenzierend ist oder nicht, ist die Frage, ob ihre Auswirkungen für den Kunden ersichtlich sind. Für Kunden ist es unbedeutend, welches Betriebssystem verwendet wird, solange die Systeme nicht ständig abstürzen. Für die Kunden ist es unbedeutend, ob *Microsoft Office* oder *OpenOffice.org* verwendet wird. Die Gründe für das eine oder das andere System mögen für Ihre Firma bedeutsam sein, für den Kunden aber sind sie irrelevant.

Um das eigene Unternehmen ansprechender für potenzielle Kunden zu gestalten, sollte man daher mehr Ressourcen in differenzierende Software investieren, die dieses Ziel unterstützt, und weniger in nicht-differenzierende. Open Source ist der Schlüssel dazu, weniger für Letztere auszugeben, indem Kosten und Risiken, die zuvor nur Ihr Unternehmen allein tragen musste, jetzt auf verschiedene, zusammenarbeitende Unternehmen aufgeteilt werden.

Dabei muss natürlich ehrlich eingeschätzt werden, welche Software tatsächlich differenzierend ist und welche nicht, was sich als schwierig herausstellt. In vielen Firmen sind Manager und Techniker nicht bereit, der Arbeit von Außenstehenden

Beachtung zu schenken, weil sie nicht glauben, dass diese genauso gut sein könnte wie die eigene. Dieses Phänomen, auch als *Not-Invented-Here-Syndrom* (NIH) bekannt, kommt einem teuer zu stehen: Ihre Mitarbeiter duplizieren die Arbeit, die andere schon geleistet haben, anstatt ihre Zeit mit der für den Betrieb so wichtigen differenzierenden Software zu verbringen.

Die Beteiligung an Open Source erhöht die Effizienz von Software-Cost-Centern, da Kosten und Risiken geteilt werden, die sie sonst alleine tragen müssten. Das Geld, welches so gespart wird, kann entweder die Gewinne erhöhen oder in andere Bereiche, wo es dringender benötigt wird, investiert werden.ihre

7 Modelle der Softwareentwicklung

Da Unternehmen nun einmal Software benötigen, muss diese auf die eine oder andere Art entwickelt werden. Zu den wichtigsten Methoden der Softwareentwicklung gehören:

- Entwicklung für den Vertrieb
- firmeninterne und Auftragsentwicklung
- Kollaboration ohne Open-Source-Lizenzierung
- Open Source

Diese Methoden unterscheiden sich in:

- ihrer Kostenverteilung
- ihrer Risikoverteilung
- ihrer Effizienz, mit der Geld in die Softwareentwicklung fließt und nicht in Gemeinkosten
- der Möglichkeit, andere von der Nutzung der Software auszuschließen

Diese Faktoren bestimmen, welche Methode für die Entwicklung einer bestimmten Software brauchbar ist. Dabei darf nicht vergessen werden, dass letztendlich der Kunde die Software finanziert, nicht der Anbieter. Der Kunde kann sein Geld in Projekten verwenden, welche eine beliebige dieser Methoden nutzen, um die benötigte Software zu beschaffen.

7.1 Entwicklung für den Vertrieb

Mit diesem Modell sind wir am vertrautesten, und das, obwohl weniger als 30 Prozent aller Software für den Vertrieb entwickelt wird.[9] Hier werden die Kosten der

9 Vgl. die Ausführungen zu *Packaged Software* unter obiger Fußnote 3.

Softwareentwicklung in der Regel von einem Hersteller allein getragen. Dieser wird versuchen, durch den Verkauf des fertigen Produkts sowohl seine Kosten zu amortisieren, als auch einen Profit zu erzielen. Dabei konzentrieren sich die Ausgaben in der Entwicklungsphase, und erst mit der Fertigstellung des Produkts kann der Hersteller anfangen, seine Kosten auszugleichen. Das Risiko eines nicht profitablen Produkts wird dabei allein vom Hersteller getragen.

Wenn sich das Produkt am Markt als erfolgreich erweist, werden die Kosten der Entwicklung nach und nach auf die Kunden verteilt. Da bei diesem Modell Kosten und Risiken nicht verteilt werden können, bis das Produkt fertig gestellt ist, benötigt der Hersteller häufig die Hilfe des Anlagemarkts, um Kosten und Risiken extern zu verteilen. Externes Kapital wird über einen langen Zeitraum hinweg benötigt, da die Entwicklung der Software und des dazugehörigen Marktes Jahre dauern kann und häufig noch mehr Zeit vergeht, bevor sich die Entwicklungskosten amortisieren und ein Profit möglich wird. Aktienmärkte erhöhen die Liquidität des Investments, da hier Investoren die Möglichkeit haben, die Einschätzung der Zukunftsaussichten einer Firma, die sich im Aktienwert widerspiegelt und nicht die tatsächlichen Umsätze des Unternehmens, in Geld zu verwandeln. Erfolgreiche Unternehmen können in die Entwicklung neuer Softwareprodukte reinvestieren, anstatt sich wieder dem Kapitalmarkt anzuvertrauen.

Die Gemeinkosten des traditionellen, nicht virtuellen Vertriebs sind extrem hoch, was dazu führt, dass weniger als 10 Prozent des Geldes, welches der Endverbraucher für ein Produkt zahlt, tatsächlich in die Vermarktung des Produkts, die Entwicklung und die Dokumentation fließen. *Microsoft* hat von seinen Verkaufseinnahmen aus dem Jahr 2005 gerade einmal 15 Prozent in die Forschung und die Entwicklung neuer Produkte investiert. Der restliche Umsatz wurde für Posten genutzt, von denen der Kunde nicht direkt profitiert, wie die sehr kostenintensive Kundenakquise für Microsoft-Produkte: Ausgaben für Werbung, Gestaltung und Herstellung attraktiver Verpackungen, die direkt nach dem Kauf entsorgt werden, Zahlungen an Einzelhändler für die Stellfläche auf dem Regal, in dem das Produkt ausgestellt wird, Verkaufspersonal und Gewinn. Die angegebenen 15 Prozent beinhalten dabei noch nicht den Preisaufschlag durch Einzel- und Großhändler; rechnet man diesen mit ein, dann sinkt der Anteil des Produktpreises, der durch die Entwicklung der Software gerechtfertigt ist, auf weit unter 10 Prozent (Microsoft Corp. 2006).

Auch auf der Erwerbsseite kommt es zu Ineffizienzen, da die gekaufte Software oft nicht dem entspricht, was der Kunde wirklich benötigt. Häufig wird Software gekauft, die sich bei genauerer Betrachtung als nicht brauchbar für die entsprechende Anwendung erweist oder nicht eingesetzt wird. Die Kosten für solche *Staubfänger* kann der Kunde häufig nicht ersetzen. Verluste kommen auch dadurch zustande, dass Softwareunternehmen Produkte aufgeben oder auslaufen lassen und nicht mehr unterstützen, die sich beim Kunden noch nicht amortisiert haben. Da selbst für Produkte, die ausgelaufen sind, ihr Quellcode nicht zur Verfügung gestellt wird, können

sie nicht weiter unterstützt werden, und dem Kunden bleibt nichts anderes übrig, als seine Investition abzuschreiben. Wenn wir all diese Risikofaktoren zusammenrechnen, dann werden, vorsichtig geschätzt, 50 Prozent (Krass 2002) der über den Handel erworbenen Software nicht genutzt oder nicht vollständig und effizient angewendet, so dass die Anschaffung ein Fehlschlag war.

Wenn wir zu dieser 50 Prozent Ausfallquote die Effizienz von weniger als 10 Prozent, mit der bei Vertriebsentwicklungen Geld in die Entwicklung reinvestiert wird, hinzurechnen, dann ergibt sich, dass gerade 5 Prozent des Geldes, welches für den Erwerb von Software ausgegeben wird, wieder in die Produktentwicklung fließt.

Das wichtigste Ergebnis dieser extrem niedrigen Effizienz ist, dass das Vertriebsmodell nur für die Entwicklung von Massenprodukten ökonomisch sinnvoll ist. Ein Massenmarkt verschleiert den Mangel an Effizienz, da jeder Kunde im Vergleich zu den Kosten für die Softwareentwicklung nur einen relativ geringen Betrag ausgeben muss. In der Gesamtsumme ergibt sich jedoch ein Betrag, der mehr als zwanzig mal so hoch ist wie die Entwicklungskosten der Software. Viele wichtige Softwareprodukte könnten für den Vertrieb überhaupt nicht geschaffen werden, da ihr Markt nicht groß genug wäre, um sowohl die Kosten für die Entwicklung als auch die hohen Gemeinkosten wieder auszugleichen. Außerdem wird ein Großteil der Produkte, obwohl sie potenziell viele Käufer finden könnten, von den Herstellern abgelehnt, weil es nicht gelingt, Unternehmen und Investoren davon zu überzeugen, dass sich ein entsprechender Markt entwickelt oder weil das Risiko zu hoch ist. Die Innovationskraft wird gedämpft, wenn ein Softwarehersteller seine Produkte über den Handel vertreibt.

Ein aufschlussreiches Beispiel für die Unfähigkeit zur Innovation ist die Tatsache, dass die wichtigsten Neuentwicklungen der weltweiten Computerbranche in den letzten zehn Jahren, Webserver und Browser, als Open-Source-Produkte im Rahmen eines von der Regierung finanzierten Universitätsprojekts entwickelt werden mussten. Keine der Firmen, denen die Fertigstellung des Produkts möglich gewesen wäre, ließ sich davon überzeugen, dass es profitabel wäre. Das einzige Unternehmen, das größere Investitionen getätigt hat, um das Internet zu entwickeln (*Autodesk*, mit der Investition in Ted Nelsons Xanadu-Projekt), entschied sich gegen die Fertigstellung des Produkts, da das Ertragsmodell, das sich *Xanadu* für das Internet vorstellte (Zahlungen an die Provider von Inhalten für jedes Wort und eine komplizierte Zuordnung von Derivaten und Referenzen) zu umständlich war. Tim Berners-Lee, letztendlich der führende Kopf bei der Entwicklung eines erfolgreichen Internets, hielt es für nicht notwendig, ein normiertes Modell zur Generierung von Einnahmen zu entwerfen und überließ es stattdessen den Nutzern, wie sie aus dem Internet Gewinne schlagen.

Wie schon zuvor erwähnt, ist Software, die wir im Geschäft erwerben, in der Regel jedem zugänglich, da sie an so viele Kunden wie möglich verkauft werden muss, um den höchstmöglichen Profit zu erreichen. Deshalb ist es schwierig oder sogar unmöglich, mit Hilfe von Software, die dem Vertriebsmodell entstammt, das Unternehmen des Kunden von anderen zu differenzieren.

7.2 Firmeninterne und Auftragsentwicklung

Während firmeninterne Entwicklungen von Programmierern durchgeführt werden, die direkt beim Kunden angestellt sind, wird eine Auftragsentwicklung von einer externen Firma übernommen, wobei der Auftraggeber exklusive Nutzrechte an dem Endprodukt erhält. In beiden Fällen werden die Programmierer üblicherweise für das Schreiben der Software bezahlt, anstatt für das fertige Produkt, wie es beim zuvor beschriebenen Vertriebsmodell der Fall wäre.

Eine andere Form der Auftragsentwicklung ist eine umfangreiche Anpassung durch den Verkäufer eines sonst frei verkäuflichen Produkts an die Wünsche eines speziellen Kunden. So könnte beispielsweise ein mittleres oder großes Unternehmen eine Standard-Webserver-Umgebung erwerben und dann den Verkäufer dafür bezahlen, das Produkt an seine speziellen Bedürfnisse anzupassen.

Sowohl im Fall firmeninterner Entwicklung als auch bei der Auftragsarbeit besitzt der Kunde die volle Kontrolle, da die Programmierer nur dann bezahlt werden, wenn sie seinen Wünschen gerecht geworden sind. In der Regel kann der Kunde auch bestimmen, ob er selbst das alleinige Nutzungsrecht für die Software erhält oder ob sie auch anderen zugänglich gemacht wird. Aufgrund dieser Kontrolle des Vertriebs durch den Kunden handelt es sich um ein ideales Modell zur Entwicklung von differenzierender Software, wenn nicht sogar um das einzige, mittels welchem Software entwickelt werden kann, die das Unternehmen des Endnutzers von anderen Unternehmen differenziert.

Es ist üblich, dass ein Auftragnehmer versucht, einen Teil der Arbeit, die er für einen bestimmten Auftraggeber durchgeführt hat, dafür zu nutzen, seine Geschäfte durch den Verkauf bereits entwickelter Produkte an andere Kunden in Schwung zu bringen. Wie erfolgreich solche Unternehmungen sind, hängt von den genauen Gegebenheiten des Vertrags und der Ehrlichkeit des Auftragnehmers ab. Wenn dieser erwarten kann, seine Arbeit auch an andere Kunden verkaufen zu können, berechnet er vielleicht einen geringeren Betrag und ein Teil der Kosten und Risiken können auf den Auftragnehmer und seinen nächsten Kunden übertragen werden. Im Falle von differenzierender Software ist dies jedoch eher ungünstig: Verkauft der Auftragnehmer die differenzierenden Merkmale eines Kunden weiter an andere Firmen, kann es schnell zu Problemen kommen.

Um in den Genuss der vollen Kontrolle zu gelangen, die mit Vertrags- oder firmeninterner Entwicklung verbunden ist, trägt der Kunde die Kosten und Risiken der Entwicklung meist allein. Für die Entwicklung von nicht-differenzierender Software gibt es daher oft kostengünstigere Möglichkeiten. Wer die vollständigen Kosten für die Entwicklung einer Software auf sich nimmt, die nur die Funktionsweise einer bereits existierenden Software kopiert, erfindet sozusagen das Rad neu und verschwendet damit Geld. Wer nicht auf die alleinige Nutzung eines Produkts und die vollständige Kontrolle über seine Entwicklung besteht, für den mag ein Open-Source-Produkt oder für den Vertrieb entwickelte Software kosteneffektiver sein.

Bei interner und Auftragsentwicklung fließt der größte Teil des investierten Geldes tatsächlich in die Entwicklung, nämlich 50–80 Prozent, verglichen mit den 10 Prozent im Falle des Verkaufs über den Handel. Die Abschläge entstehen hier durch die Kosten für die Kundenakquise des Auftragnehmers, die Kosten der Bereitstellung nicht dauerhaft genutzter Fachkompetenzen und durch die Gewinnmargen des Auftragnehmers.

Häufig verfehlen firmeninterne und Auftragsentwicklung das Ziel einer funktionierenden Software, die im Unternehmen des Kunden eingesetzt werden kann und die geforderten Ansprüche erfüllt. Diesen Entwicklungsmodellen kann eine Erfolgsrate von 50 Prozent (U.S. General Accountion Office 1997), ähnlich dem Vertriebsmodell, bescheinigt werden.

7.3 Kollaboration ohne Open-Source-Lizenzierung

Firmenübergreifende Zusammenarbeit in der Softwareentwicklung fand in der Vergangenheit üblicherweise im Rahmen von Konsortien statt, wobei es nicht selten zu gigantischen Fehlschlägen kam. In jüngerer Zeit wurden milliardenschwere geschlossene Konsortien häufig durch erfolgreichere Open-Source-Projekte ersetzt. Man erinnere sich an *Taligent* und *Monterey*, zwei Konsortien, die zu dem Zweck gebildet wurden, einen UNIX-Ersatz zu schaffen. *Linux* ersetzte sie. Oder das Common-Desktop–Environment-Projekt, welches in den meisten Firmen, die vorher *CDE* unterstützten, durch den Open-Source-Desktop *GNOME* ersetzt wurde.

Vor einigen Jahren gründete das große Landwirtschaftsunternehmen *Cargill* ein Konsortium in Form einer so genannten *gated community*: Ziel war es, die eigenen Partner in den Genuss der Vorteile von Open Source kommen zu lassen, jedoch gleichzeitig für Geheimhaltung zu sorgen und die Vorteile der Software nur mit den Mitgliedern des Konsortiums zu teilen. Zwei Jahre später verabschiedete *Cargill* sich von dem Projekt. Nicht-differenzierende Software in einem geschlossenen Konsortium zu entwickeln, ist einfach nicht angemessen. Wo es nicht um den Schutz differenzierender Merkmale geht, ist es sinnvoll, die Türen weit zu öffnen und Mitglieder zuzulassen, die nützliche Beiträge leisten können, auch wenn sie sich finanziell nicht beteiligen können. Ein Konsortium ist kostenintensiver, da weniger Mitglieder die Kosten und Risiken unter sich aufteilen als bei einem Open-Source-Projekt und das im Rahmen eines aufwändigeren Gefüges mit höheren Gemeinkosten, als es in einem entsprechenden Open-Source-Projekt notwendig wäre. Geschlossene Konsortien vertreten im Allgemeinen das Prinzip *pay-for-say*, d. h. nur wer sich finanziell beteiligt, darf auch mitentscheiden; Open Source dagegen setzt auf fachliche Leistung. Im Falle von *pay-for-say* kann ein Mitglied auf eine Schädigung des gesamten Projekts hinarbeiten, wenn dies zu seinem eigenen Vorteil ist. Produktplanung in Konsortien führt häufig zu unlösbaren Differenzen zwischen den beteiligten Firmen, da diese unterschiedliche Vorstellungen vom Vertrieb eines Produktes haben und Unstimmigkeiten zwischen Unternehmen im Marketing subjektiv und schwer zu lösen sind.

Betrachtet man die zahlreichen Fehlschläge von Konsortien und auf der anderen Seite die hohe Erfolgsquote großer Open-Source-Projekte zwischen denselben Unternehmen, lässt sich schließen, dass die Fairness, die mit Open-Source-Lizenzerteilung einhergeht, ein entscheidender Bestandteil effektiver Kollaboration zwischen einer großen Zahl von Teilnehmern mit unterschiedlichen Interessen ist.

7.4 Open Source

Das Open-Source-Modell beinhaltet die Zusammenarbeit zahlreicher Beteiligter (Individuen, Unternehmen, akademische Institutionen und andere) bei der Entwicklung eines Softwareprodukts.

Normalerweise wird die initiale Entwicklung von einer einzelnen Partei durchgeführt, wie es auch bei der innerbetrieblichen Entwicklung oder der Auftragsentwicklung der Fall ist. Die Software wird der Öffentlichkeit verfügbar gemacht, sobald sie für andere nutzbringend ist, aber im Allgemeinen vor der endgültigen Fertigstellung, und damit lange bevor ein Produkt im Handel erhältlich wäre. Sobald die Software nutzbar ist, machen andere Parteien sie sich zunutze. Und auch erst in diesem Moment, wenn das Produkt anderen einen Nutzen bietet, funktioniert das Open-Source-Modell wirklich, da erst jetzt auch eine Motivation besteht, die Software um zusätzliche Eigenschaften zu ergänzen, die für den Einzelnen interessant sind. Diese Erweiterung geschieht durch die Angestellten eines Kunden oder durch Auftragnehmer in dessen Kontrollbereich.

Die inkrementellen Kosten für das Hinzufügen eines Features sind dabei viel geringer als die Kosten für eine vollständige Entwicklung. Parteien, die Modifizierungen vornehmen, haben ein Interesse daran, diese so zu gestalten, dass sie von den anderen Entwicklern, die an dem Projekt beteiligt sind, akzeptiert werden und so in den Kern des Quellcodes eingearbeitet werden, den alle Entwickler teilen. Erfolgt diese Einbindung nicht, ist es kostenintensiver, die hinzugefügten Features aufrechtzuerhalten, da es Aufgabe der entwickelnden Partei bleibt, Änderungen am Hauptquellcode zu verfolgen und die Kompatibilität mit der sich verändernden *Codebase*, der Gesamtheit des Quellcodes, aufrechtzuerhalten.

Open Source führt daher häufig zur Bildung einer Gemeinschaft von Entwicklern, welche dazu beitragen, ein nützliches Produkt zu schaffen. Die Kosten und Risiken der Entwicklung werden unter den Programmierern aufgeteilt, und sollte jemand das Projekt verlassen, so können es die Übrigen fortführen. Die Aufteilung von Kosten und Risiken beginnt in dem Moment, an dem das Projekt reif genug ist, dass sich eine Community außerhalb des initialen Entwicklerkreises bildet.

Open-Source-Software wird direkt von ihren Endnutzern entwickelt. So werden zum Beispiel Features des Apache-Webservers oft von den Unternehmen hinzugefügt, die diese für ihre eigenen Websites benötigen, oder das Untermehmen beauftragt Dritte, die entsprechenden Erweiterungen vorzunehmen.

Die Kunden eines bestimmten Open-Source-Produktes geben sich in der Regel zu erkennen: Sie suchen in einem Open-Source-Software-Verzeichnis nach dem Produkt, laden es herunter und testen die Software. Wenn diese Tests erfolgreich verlaufen, setzen sie die Software für sich ein. Sie entwickeln somit ein fortdauerndes Interesse an dem Produkt. Wenn sie in dieser Situation eine Erweiterung benötigen, besteht ein Anreiz, sich an der weiteren Entwicklung der genutzten Software zu beteiligen.

Unternehmen, die sich an Open-Source-Kollaborationen beteiligen, möchten die jeweilige Software nicht-differenzierend und im Sinne eines Kostenzentrums nutzen. Für sie ist es nicht von Bedeutung, dass Open Source an sich keinen Profit schafft. Durch die Software sind sie in der Lage, in anderen Bereichen Profite zu machen, und damit dies so bleibt, müssen sie in ihre Kostenzentren investieren. Im Falle von differenzierender Software bleibt ihnen wenig anderes übrig, als das Produkt in Form einer firmeninternen oder einer Auftragsentwicklung zu erstellen, damit die differenzierenden Merkmale nicht von Konkurrenten genutzt werden können. Dagegen haben sie im Falle von nicht-differenzierender Software die Wahl zwischen einem Vertriebsprodukt und Open Source. Es stellt sich die Frage, was wirtschaftlicher ist.

Da sich die Kunden zu erkennen geben, leidet Open Source nicht an der Ineffizienz einer Vertriebsentwicklung, welche Werbung oder andere kostenintensive Mechanismen zur Kundengewinnung mit sich bringt. Auch ist Open Source wenigstens genauso effizient, wie es eine Eigen- oder Auftragsentwicklung wäre, wenn es darum geht, das zur Verfügung stehende Geld in die Softwareentwicklung anstatt in Gemeinkosten fließen zu lassen.

Aufgrund der höheren Effizienz in diesem Bereich können mit Hilfe von Open Source selbst Produkte entwickelt werden, die keinen Massenmarkt ansprechen und daher nicht wirtschaftlich erfolgreich im Rahmen des Vertriebsmodells entwickelt werden könnten.

Craig Small, Ingenieur und Entwickler von Netzwerk-Infrastrukturen, fasst die Vorteile für seine Kunden wie folgt zusammen:

> „Durch die Nutzung von Open Source haben sich die Initialkosten für Software drastisch reduzieren lassen. Dies war deshalb so wichtig, da sich so das Interesse dieses bestimmten Kunden wecken ließ. Ich hatte geplant, das Managementsystem des Netzwerks von Grund auf neu zu entwickeln, als ich auf ein Projekt stieß, das nahe genug an dem dran war, was ich benötigte. Ich war in der Lage, für das Problem nur einen Bruchteil der Arbeitsstunden aufzuwenden, die ich benötigt hätte, wenn ich von Null hätte anfangen müssen.
>
> Die Erweiterung und Anpassung von Open-Source-Software ist wesentlich einfacher, da ihre Entwickler von vorneherein davon ausgehen, dass andere ihre Arbeit auf eine Art ausbauen werden, die sie selbst nicht eingeplant hatten. Daher schreiben sie so, dass Ergänzungen einfacher

einzubinden sind. Die Programmierer proprietärer Software schreiben, als ob niemand außerhalb der eigenen Firma jemals ihren Quellcode lesen wird. Wir begannen mit einem Open-Source-Projekt, das für eine Hardware gedacht war, die völlig anders war als unsere eigene, konnten es aber leicht so abändern, dass sich unsere Anlagen überwachen ließen. Ich habe meine Erweiterungen wiederum dem Open-Source-Projekt zur Verfügung gestellt. So kann ich darauf vertrauen, dass für den Fall, dass ich die Firma verlasse, sich dennoch jemand, zumindest ein Teilnehmer des Projekts, weiterhin meiner Erweiterungen annimmt.

Inzwischen existiert eine ganze Gemeinschaft von Menschen, sowohl Entwicklern als auch nicht-programmierenden Nutzern, mit denselben technischen Bedürfnissen, die sich um das Softwareprojekt, zu dem wir alle beigetragen haben, entwickelt hat. Wir befinden uns im Austausch darüber, wie wir unsere Bedürfnisse besser erfüllen können und tragen so Ideen zusammen, auf die wir alleine nie gekommen wären.

Proprietäre Software birgt Kosten in sich, über die Unternehmen erst seit kurzer Zeit nachdenken. Organisationen wie die *Business Software Alliance* haben mit der Bundespolizei Razzien durchgeführt, bei denen sie von Unternehmen schriftliche Nachweise darüber forderten, dass sie für jede einzelne Software auf ihrem Gelände eine Lizenz erworben hatten. Sehr viel Zeit und Mühe wird dafür aufgebracht, sicherzustellen, dass sich Unternehmen an immer lästigere Lizenzbestimmungen halten. Warum ist es mir rechtlich gesehen nicht möglich, mein *Windows XP* von meinem Laptop auf meinen Desktop zu übertragen? Durch Open Source wird solch unproduktiver Nonsens vermieden."

Nicht alle Open-Source-Projekte sind erfolgreich. Viele scheitern bereits früh oder bleiben Solo-Projekte, die nur geringe Ressourcen in Anspruch nehmen. Die Ausgaben für solche Projekte sind jedoch unbedeutend, da sie eingestellt werden, bevor sie erste bedeutende Kunden oder eine Gemeinde von Programmierern anziehen. Reifere Open-Source-Projekte können eingestellt werden, wenn sich eine bessere Alternative ergibt, aber dabei ist es häufig möglich, den Quellcodes, die Daten und Erkenntnisse eines Projekts für ein anderes zu verwenden, und so gehen die Investitionen in die Entwicklung des Projekts nicht verloren.

Die Kosten der Teilnahme an fortgeschrittenen Open-Source-Projekten unterscheiden sich stark von den Kosten einer Vertriebsentwicklung bzw. interner oder Auftragsentwicklung. Der größte Teil der Kosten entsteht aufgrund des Zeitaufwands, den Mitarbeiter auf die Teilnahme am Projekt verwenden. Dieser Posten setzt sich zusammen aus den Personalkosten der Softwareevaluation, den Kosten für eigenes Personal oder Auftragnehmer, die eine existierende Open-Source-Software den Kundenbedürfnissen anpassen, und den Kosten für den Support der Software für interne Nutzer. Außerdem besteht immer die Möglichkeit, dass Zeit in eine Software inves-

tiert wird, welche dann doch ersetzt wird, da sie die jeweiligen Bedürfnisse doch nicht den Erwartungen entsprechend erfüllen kann.[10] Die maximalen Kosten für Open Source würden sich ergeben, wenn keine Gemeinschaft außerhalb des Kunden selbst bestünde: Die Kosten wären dann vergleichbar mit denen einer Auftragsentwicklung oder einer betriebsinternen Entwicklung, bei welcher der einzelne Kunde die Gesamtkosten trägt. Die tatsächlichen Kosten sind geringer, je nachdem wieviele Teilnehmer aktiv mitarbeiten und je nachdem wieviel Arbeit notwendig ist. Die Investition, die man verliert, liegt in der Hauptsache in den aufgewandten Personalkosten. Das mit eingerechnet, ergibt sich durch Open Source eine wirtschaftliche Effizienz, die mindestens so hoch ist, wie die einer firmeninternen oder Auftragsentwicklung und viel höher als die einer Entwicklung zu Vertriebszwecken.

7.5 Zusammenfassung der Methoden der Softwareentwicklung

Open Source hat mehrere entscheidende ökonomische Vorteile gegenüber allen anderen Methoden der Softwareentwicklung: Hier verbinden sich Effizienz in der Zuweisung von Ressourcen mit einer besseren Kosten- und Risikenverteilung als bei jeder anderen gängigen Methode der Softwareentwicklung. So können Produkte entwickelt werden, die im Vertrieb nicht kosteneffizient vertrieben werden könnten, weil es für sie keinen genügend großen Markt gibt. Bei Open Source können die Kosten und Risiken von Projekten verteilt werden, während bei einer Entwicklung durch die eigene Firma oder einen Auftragnehmer eine Partei alle Kosten und Risiken tragen muss. Open Source verteilt Kosten und Risiken direkt, ohne sich an den Anlagemarkt wenden zu müssen. Die Verteilung von Kosten und Risiken beginnt viel früher als es beim Vertriebsmodell der Fall wäre. Open Source gibt dem Kunden die volle Kontrolle über die Anpassung des Produktes. Der Kunde hat jedoch keine Kontrolle darüber, wer Zugang zu dem Produkt hat. Aus diesem Grund ist Open Source in den meisten Fällen keine gute Wahl, wenn es um die Herstellung differenzierender Software geht.

Für Unternehmen, die nicht-differenzierende Software benötigen, im Grunde genommen für alle Firmen, ist es sehr ratsam, einen Teil der Entwicklungsarbeit mit Hilfe von Open Source zu leisten, um so eine höhere ökonomische Effizienz zu erreichen. Die Teilnahme an Open-Source-Entwicklungsgemeinschaften sollte einen wichtigen Teil einer Gesamtstrategie eines jeden Unternehmens darstellen, die darin besteht, weniger Geld in nicht-differenzierende Software zu investieren und dafür mehr in die Entwicklung differenzierender Software.

10 Auch im Rahmen des Vertriebsmodells entwickelte Software beinhaltet Teilnahmekosten. Teilweise entsprechen diese den oben im Zusammenhang mit dem „Staubfänger"-Problem erwähnten Kosten. Keine Erwähnung finden die Kosten, die nach dem Erwerb entstehen. Hier finden sich unter anderem die Kosten für die Dokumentation durch den Händler (häufig eine Vollzeitaufgabe für die Programmierer) und die Kosten für die häufig langwierigen Verhandlungen mit der Serviceabteilung des Händlers, bis ein bestimmtes Problem behoben ist.

8 Wer trägt zu Open Source bei und wie finanziert sich dieser Beitrag?

Die Open-Source-Teilnehmer können folgendermaßen kategorisiert werden:

- Freiwillig Mitwirkende
- Anbieter von Linux-Distributionen
- Unternehmen mit nur einem Open-Source-Programm als ihrem Hauptprodukt
- Unternehmen, denen Open-Source-Software den Verkauf von Hardware und Lösungen ermöglicht
- Endanwender-orientierte Unternehmen und deren Auftragnehmer
- Dienstleistungsunternehmen
- Staatliche Institutionen und Verwaltungen
- Akademiker und Wissenschaftler

Freiwillig Mitwirkende

Ich (Bruce Perens) war selbst ein Beispiel für diese Form der Mitarbeit, als ich mich von 1993 bis 1998 intensiv mit der Entwicklung von Open Source beschäftigt habe, obwohl dies in keinerlei Verbindung zu meiner beruflichen Tätigkeit stand.

Die FLOSS-Studie des *International Institute of Infonomics* (2002) beschrieb die Entwicklung von Open Source zu jener Zeit in erster Linie als eine Art Freizeitbeschäftigung. Die Mitarbeit von Privatpersonen, die nicht direkt aufgrund einer primär finanziellen Motivation an der Entwicklung von Open-Source-Software mitwirken, ist beträchtlich genug, um über den Begriff des Hobbys hinauszugehen. Angesichts der offenkundigen Professionalität ihrer Arbeit[11] und der Tatsache, dass sie sich nicht aus pekuniären Gründen am Projekt beteiligen, ist freiwillig Mitwirkender vielleicht eine treffendere Beschreibung.

Schon Eric Raymond hat darauf aufmerksam gemacht, dass die Motivation der freiwilligen Mitarbeiter in erster Linie immaterieller Natur ist und dass ein wichtiger Beweggrund für ihre Beteiligung die Mitgliedschaft in einer Gemeinschaft gegenseitigen Respekts ist, in der die Entwickler von ihren Mitstreitern für die Qualität und den innovativen Charakter ihrer Arbeit Anerkennung erhalten. In der FLOSS-Studie wurden Open-Source-Entwickler hinsichtlich ihrer Motivation befragt, und so wurde herausgefunden, dass sich viele aus reiner Wissbegier und dem Bestreben, ihre Kenntnisse zu erweitern, engagieren. Ich denke, dass ihre Beweggründe mit denen eines

11 Eine Studie von *Coverity* in Zusammenarbeit mit der *Stanford University* (Kerner 2006) zeigt, dass die am häufigsten verwendeten Open-Source-Anwendungen 0,434 Bugs pro 1 000 Zeilen Code beinhalten, während nach Angaben der *Carnegie Mellon University* (Delio 2004) kommerzielle Nicht-Open–Source-Software eine Fehlerhäufigkeit von 20–30 Bugs pro 1 000 Zeilen Code aufweist.

Künstlers vergleichbar sind: Ebenso wie der Antrieb eines Malers sich darin manifestiert, dass die Menschen seine Gemälde bewundern, wünscht ein Programmierer sich Nutzer, die seine Software zu schätzen wissen.

Mit der zunehmenden Nutzung von Open Source innerhalb von Firmen, die sich dann im Laufe der Zeit für deren Weiterentwicklung engagieren, kommt es zu einem stetigen Wandel vom freiwillig zum professionell Mitwirkenden. Die einst freiwillig Mitwirkenden werden von Firmen angestellt, die ihre Mitarbeit an Open-Source-Projekten in der Arbeitszeit unterstützen. In dem Moment, da Arbeitgeber gegenüber freiwillig Mitwirkenden ein Interesse an Open Source bekunden, treten diese aus „ihrem stillen Kämmerlein" und werden zu Experten innerhalb des Unternehmens.

8.1 Anbieter von Linux-Distributionen

Red Hat und *Novell* sind als Distributoren Linux-basierter Systeme bekannt. Erstaunlicherweise wird die Mehrheit der Open-Source-Software jedoch nicht von Firmen, die Open-Source-Software als Hauptprodukt verkaufen, entwickelt. Sie verwerten im Wesentlichen nur die Arbeit anderer. Sie übernehmen Wartungsarbeiten, um Bugs für ihre zahlenden Kunden zu entfernen und sie beteiligen sich dann an der Entwicklung von Open-Source-Software, wenn dies für ihr Produkt oder die Erschließung eines neuen Marktes erforderlich ist. Bei den Linux-Distributionen handelt es sich meist um Unternehmen mittlerer Größe, deren Arbeitnehmer nur einem kleinen, wenngleich sehr aktiven Kreis der Open-Source-Gemeinschaft angehören. Gelegentlich übertreiben sie im Rahmen ihres Marketings ihre Bedeutung für die Entwicklung. Beispielsweise wirbt eine Linux-Distribution damit, 300 Programmierer zu beschäftigen, jedoch beteiligt sich nur ein viel geringerer Teil regelmäßig an der Entwicklung von Open-Source-Software. Der Großteil scheint sich vielmehr mit der „Vertriebstechnik" oder der Erarbeitung interner Lösungen für Kunden zu befassen.

Unter dem wirtschaftlichen Aspekt betrachtet, scheint Open Source für Linux–Distributionen eigentlich am wenigsten geeignet zu sein. Wenn man die Merkmale offenlegt, mit denen man sich von anderen Firmen differenziert, ermöglicht man der Konkurrenz, sich diese anzueignen und damit die Produktdifferenzierung zu schmälern. Genau dies ist denn auch die missliche Lage, in der sich die Linux-Distributionen befinden: Ihre Produkte sind hauptsächlich Open Source, die Kunden wünschen auch, dass es so bleibt, und so haben sie Schwierigkeiten zu erreichen, dass sich ihr Produkt von den anderen abhebt, insbesondere, da der Kunde weiß, dass das Produkt anderswo kostenlos angeboten wird.

Linux-Distributionen versuchten ursprünglich, dem Problem der Produktdifferenzierung mit den Open-Source-Verkaufsstrategien der ersten Generation beizukommen, die von Eric Raymond in „The Cathedral and the Bazaar" erläutert werden. Die Mehrzahl dieser Geschäftsmodelle sah eine Kombination von Open-Source-Software mit einem anderen Produkt als Einnahmequelle vor: Dienstleistungen für das Open-Source-Produkt oder eine proprietäre Softwareerweiterung der Open-Source–

Komponente. Die Open-Source-Software wäre hierbei die Geschäftsgrundlage des Unternehmens, die Einnahmen würden allerdings über andere Produkte generiert werden, wobei diese mit der Open-Source-Software unmittelbar im Zusammenhang stünden. Leider waren Dienstleistungen in der Zeit der *early adopters* für *Linux*[12] allein nicht ausreichend, um eine entsprechende Profitabilität der Distributionen zu erzielen.

Heute erfahren wir die zweite Generation der Open-Source-Verkaufsstrategien. Einige der Linux-Distributionen sind proprietärer Software nachempfunden. Hinter ihrem kostenpflichtigen Softwarepaket oder ihrer Per-Platz-Lizenzen verbirgt sich ein Produkt, das der Kunde anderswo kostenfrei bekommen könnte. Damit dieses Prinzip überhaupt funktioniert, wurden verschiedene Strategien miteinander kombiniert: Zunächst erfolgt die Entwicklung einer Marke, der mehr Vertrauen und Wert zugeschrieben wird als der Software an sich. Der Verkäufer investiert nun in die Zertifizierung des Produkts durch Anbieter proprietärer Anwendungen, von denen jeder nur für einige der Distributionen Supportleistungen zur Verfügung stellt.[13] Kunden, die Support für eines der proprietären Produkte auf Linux-Basis brauchen, wird so ein Anreiz gegeben, sich die Linux-Version dieses Anbieters zu kaufen. Einige dieser Supportleistungen sind allerdings nur eingeschränkt nutzbar, wie z. B. Berichte zu Sicherheitsproblemen und Bug-Patches, die nur für diejenigen verfügbar sind, die sich das kostenpflichtige Softwarepaket und die Einzelplatzlizenzen angeschafft haben. Wenn ein Kunde die freie Software auf mehr Systeme laden sollte, als er bezahlt hat und der Verkäufer das herausfindet, dann erfolgt die Aufhebung seines Wartungsvertrages und es werden ihm Sicherheitsinformationen vorenthalten, die wichtig für den weiteren reibungslosen Betrieb seiner Software wären. Die passendste Bezeichnung für dieses Geschäftsmodell ist daher wahrscheinlich proprietäre Open Source, bei der Support- und Serviceleistungen angeboten werden, das Geschäft jedoch im proprietären Bereich operiert. Dieses Geschäftsmodell widerspricht der eigentlichen Idee von Open Source und steht daher in antagonistischer Art und Weise dem Beitrag der freiwillig Mitwirkenden, die den Großteil der Open-Source-Software entwickelt haben, gegenüber. Diese hatten nicht die Absicht, ein zweites *Microsoft* zu schaffen und sperren sich folglich gegen die Beschlagnahmung von Serviceinformation ihrer Software. Solches steht in klarem Widerspruch zum Geiste, wenn auch nicht zum Wortlaut, von Lizenzmodellen wie der *GPL*. Generell unterstützen die freiwillig Mitwirkenden genau jene Unternehmen, von denen sie auch etwas halten. Als interne

12 Der Begriff *early adopter* (englisch für: frühzeitiger Anwender, Visionär) bezeichnet eine Firma, die frühzeitig die neusten technischen Errungenschaften oder die neusten Varianten von Produkten erwirbt. Die Zeit der *early adopters* ist Bestandteil der Diffusionstheorie und stellt die zweite Phase der Marktakzeptanz eines innovativen Produkts dar (Anm. d. Red.).

13 Die *Linux Standard Base* war ursprünglich dafür gedacht, proprietären Anwendungsanbietern die Möglichkeit zu geben, ihre Produkte für alle standardisierten Linux-Distributionen anzubieten. Sie ist inzwischen von der *ISO* als internationaler Standard akzeptiert worden und verschiedene prominente Linux-Distributionen (*RedHat*, *SUSE*, *Ubuntu* u. a.) bemühen sich um Zertifizierung.

Open-Source-Experten ihrer Firma empfehlen sie nur die Firmen weiter, die sie für gut befinden. Somit steht das Geschäftsmodell der proprietären Open Source vor nicht unbedeutenden Herausforderungen.

An dieser Stelle ist es wichtig, hervorzuheben, dass sich nicht alle Linux-Distributionen an proprietärer Open Source beteiligen und dass proprietäre Open Source bei der Entwicklung von Open-Source-Software nicht das übliche Geschäftsmodell ist. Die überwiegende Mehrheit der Mitwirkenden an Open Source gehört einer der anderen der hier untersuchten Kategorien an.

Das Geld für den Kauf von Linux-Distributions-Lizenzen und den damit verbundenen Support- und Serviceleistungen stammt aus den IT-Kostenstellen der Unternehmen.

8.2 Unternehmen mit nur einem Open-Source-Programm als ihrem Hauptprodukt

Diese Unternehmensform kann in verschiedene Modelle unterteilt werden:

- Modell der gemischten Lizenzierung: Open Source und proprietär
- Ein zentrales Open-Source-Programm in Verbindung mit proprietären Software-Accessoires
- Reines Open-Source-Modell mit den dazugehörigen Serviceleistungen

Im Folgenden wird jedes Modell näher betrachtet.

Modell der gemischten Lizenzierung: Open Source und proprietär

Beispiele hierfür sind Firmen wie *MySQL AB* und *Asterix PBX*, die Datenbanken erstellen, oder *Trolltech*, von denen das Qt-Toolkit für Programmierung graphischer Benutzerschnittstellen (GUI) produziert wird. Bei dieser Unternehmensform wird dieselbe Software unter zwei verschiedenen Lizenzen vertrieben: einer Open-Source–Lizenz und einer kommerziellen Lizenz.

Die gewählte Open-Source-Lizenz, häufig die *GPL*, enthält dabei eine „bittere Pille“, welche die Herstellung von proprietären Folgeentwicklungen zu kommerziellen Zwecken verhindern soll. Da die *GPL* vorschreibt, dass Derivate generell nur als Quellcode im Rahmen der *GPL* oder einer zur *GPL* kompatiblen Lizenz vertrieben werden dürfen, würde man die Produktdifferenzierung jeder Software unterbinden, auf die dieses zutrifft.

Um dem aus dem Weg zu gehen und eine Produktdifferenzierung zu erhalten, muss der Hersteller proprietärer Derivate eine kommerzielle Lizenz für das gleiche Produkt erwerben. Dies verschafft dem Entwickler der Open-Source-Software ein direktes Einkommen für den Einzelverkauf von Software an die Hersteller solcher proprietärer Folgeentwicklungen.

Dieses Modell funktioniert allerdings nur bei Software, die in solche abgeleiteten Arbeiten integriert wird, etwa bei Softwarebibliotheken. Für Anwendungen ist dieses Modell im Allgemeinen unbrauchbar.

Die Zukunft dieses Modells ist jedoch fraglich, da ein Programmierer eine Softwarebibliothek als „Server" erstellen und alle darin enthaltenden Funktionen in ein anderes Programm exportieren kann, ohne dabei etwas zu schaffen, was nach dem Urheberrecht oder den Definitionen der Open-Source-Lizenzen als Derivat gelten würde. In der UNIX-Sprache werden solche Server als *daemons* bezeichnet, weshalb die Einbettung von Software in einen *daemon*, um die Herstellung eines Derivats zu vermeiden, als *daemonization* bezeichnet wird. Möglicherweise wird dieser Vorgehensweise durch zukünftige Open-Source-Lizenzen ein Riegel vorgeschoben.

Heutzutage kann man den MySQL-Datenbankserver in einer proprietären Anwendung nutzen, ohne über eine entsprechende Handelslizenz zu verfügen. Hierfür muss man lediglich die *MySQL Database Engine* als Server verwenden (Standardmodus) und sich einer speziellen Variante der *MySQL Client Library* bedienen, deren Lizenzvereinbarung für proprietäre Anwendungen geeignet ist. Eine solche Bibliothek wird von *MySQL AB* selbstverständlich nicht unterstützt.

Solcherart orientierte Unternehmen steigern üblicherweise ihre Einkünfte aus der kommerziellen Lizenzierung durch zusätzliche Einnahmen aus Schulungen und Softwareentwicklungen.

Die Hauptkunden von *MySQL* sind gewerbliche Nutzer. Trolltech zählt in der Hauptsache Anwendungsentwickler und Entwickler aus dem Embedded-Devices–Bereich zu seinen Kunden. Die Geldmittel für den Erwerb derartiger Produkte kommen zumeist aus den Budgets der IT- und Entwicklungsabteilungen.

Ein zentrales Open-Source-Programm in Verbindung mit proprietären Software-Accessoires

Eric Raymond nennt dieses Modell *widget frosting* und hat es ausführlich beschrieben. *Sendmail Inc.* ist ein Beispiel dafür. *Sendmail* hat eine Serie proprietärer Produkte rund um den Open-Source-E-Mail-Server *Sendmail* entwickelt. Über diese werden die anfallenden Pflege- und Wartungsarbeiten des zentralen Open-Source-Produkts finanziert. Einige Anbieter von Linux-Distributionen beabsichtigen, auf diesem Modell zu operieren, und so mancher IBM-Geschäftsbereich verwendet es bereits: z. B. der Vertrieb der proprietären DB2-Datenbank für *Linux*. Diese Unternehmen arbeiten im Grunde genau wie die Firmen, die proprietäre Software auf den Markt bringen. Die Geldmittel für den Erwerb der Produkte entstammen den Budgets der IT-Abteilungen.

Reines Open-Source-Modell mit den dazugehörigen Serviceleistungen

Dieses Modell wurde in Eric Raymonds Artikel als eines der wichtigsten beschrieben, hat sich aber bisher als weniger erfolgreich als erwartet erwiesen. Viele Open-Source–Entwickler bessern ihre Einkünfte auf, indem sie Serviceleistungen für die von ihnen entwickelten Programme anbieten. Auch eine Anzahl neuer Firmen hat dieses Modell bereits adaptiert, einige unter Verwendung erheblichen Risikokapitals. Viele dieser neuen Firmen scheinen ein Zertifizierungsmodell zu verwenden, bei dem sie Dienstleistungen für eine bestimmte, zertifizierte Version der Software anbieten und das vielleicht mit dem als proprietäre Open Source bezeichneten Geschäftsmodell einiger Linux-Distributoren vergleichbar ist.

Einige wenige der klein- und mittelständischen Unternehmen haben es geschafft, nachhaltige Einnahmen zu erzielen mit der Entwicklung von Open-Source-Software als ihrem zentralen Tätigkeitsbereich und der Serviceleistung für diese Software als ihrem Ertragszentrum. Viele sind jedoch auch spektakulär gescheitert, etwa *Linuxcare* (heute *Levanta*). Firmen, die eigentlich potenzielle Servicekunden wären, scheinen es vorzuziehen, ihren Open-Source-Support intern zu regeln bzw. über einen Anbieter abzuwickeln, zu dem sie bereits Geschäftsbeziehungen unterhalten oder der in der Lage ist, Serviceleistungen für mehr als ein Programm zur Verfügung zu stellen. Darüber hinaus kann hier aber auch ein Early-Adopter-Problem vorliegen. Solche Erstanwender lehnen es im Allgemeinen ab, von einem Serviceunternehmen „bemuttert" zu werden. Es wird sich erst noch zeigen, ob dieses Modell effektive Ergebnisse erzielen kann.

Die Geldmittel für die Serviceleistungen, die von einem solchen Unternehmen zur Verfügung gestellt werden, stammen auch hier aus den Budgets der IT-Abteilungen.

8.3 Unternehmen, denen Open-Source-Software den Verkauf von Hardware und Lösungen ermöglicht

IBM und *HP* sind beispielsweise solche Unternehmen. Hardware lässt sich großartig in Verbindung mit Open-Source-Software verkaufen. Es kostet bekanntlich fast nichts, Software zu kopieren, jedoch kann man ein Brot nicht vervielfachen, ohne zumindest ein Pfund Mehl zu verbrauchen. Solange wir nicht im Besitz des „Star–Trek-Replikators"[14] sind, wird Hardware ein schwer zu vervielfältigendes Produkt bleiben. Wenn man dem Kunden Einblick in die Interna eines Hardwareprodukts gewährt, bedeutet dies nicht zwangsläufig, dass man dessen Produktdifferenzierung dadurch aufhebt, wie das vielleicht bei Software der Fall wäre. Die Hersteller von Hardware, die bei der Entwicklung von Open Source mitwirken, tun dies, um den Absatz

14 Der *Replikator* ist ein fiktionales Gerät, der viele verschiedene physische Objekte auf Abruf produzieren kann und dabei vermutlich auf in einem Computer abgelegte Designs zurückgreift. In der Serie *Star Trek: The Next Generation* trat eine Person an das Gerät heran und gab ihm eine verbale Anweisung: „tea, earl grey, hot." Eine Tasse und der Tee wurden im Bruchteil einer Sekunde produziert und bereitgestellt.

ihrer Hardwareprodukte zu fördern. Denn Hardware ist ohne die entsprechende Software nutzlos und dies gilt insbesondere für Computer, die ohne ein Betriebssystem, das Computerhardware und Softwareanwendungen miteinander verbindet, nicht zu gebrauchen wären. Die Entwickler von Open-Source-Software scheinen vor allem auf dem Gebiet der Systemprogrammierung besser zu sein als auf jedem anderen Gebiet der Programmierung. Der Linux-OS-Kernel ist inzwischen mindestens genauso gut, wenn nicht besser, als viele proprietäre Betriebssysteme für vergleichbare Hardware. Die Hersteller von Hardware haben früher Milliarden für proprietäre Betriebssysteme ausgegeben, die für sie immer eher technologische Notwendigkeit als Ertragszentrum waren. Die Gewinnspannen wurden durch die Hardware bestimmt. Viele der Hersteller haben *Linux* bereitwillig übernommen, weil es ihnen die Verteilung von Kosten und Risiken des Betriebssystems auf mehrere Firmen ermöglicht, die Rentabilität zudem größer ist als bei vergleichbaren proprietären Betriebssystemen und es obendrein bei den Kunden hoch im Kurs steht.

Hardwareunternehmen eignen sich sehr gut als Open-Source-Produzenten, weil Open Source und Produktdifferenzierung in diesem Falle nicht so stark miteinander in Konflikt stehen wie bei Softwareunternehmen. Softwareprodukte offenzulegen, ermöglicht diesen Unternehmen den Vertrieb, mindert nicht im Geringsten die Differenzierung ihrer Hardware-basierten Produkte und kann somit ihr Ertragszentrum nicht negativ beeinflussen.

Die Erlöse aus dem Hardwarevertrieb werden bei dieser Art von Unternehmen üblicherweise durch Serviceleistungen und mitunter auch Schulungen erhöht. Die finanziellen Mittel für die Gesamtheit der angebotenen Hardware und Serviceleistungen stammen aus den jeweiligen Hardwarebudgets der IT-Abteilungen.

8.4 Endanwender-orientierte Unternehmen und deren Auftragnehmer

eBay ist ein Beispiel für diese Art der Beteiligung. Open-Source-Software wird von vielen Firmen für ihre eigenen Arbeitsabläufe genutzt, wobei Webapplikationen, neben vielen anderen, einen besonderen Stellenwert haben. Diese Unternehmen bilden einen signifikanten Teil der Mitwirkenden an Open Source. Ihre Beiträge stammen im Allgemeinen von den Mitarbeitern des Softwaresupports und der Softwareentwicklung der Firmen oder aber ihren Dienstleistern, welche die Open-Source-Software den Bedürfnissen des Unternehmens entsprechend anpassen. Das Interessante an diesen Unternehmen ist, dass sie im Grunde genommen die Hauptklientel aller anderen Unternehmen, die zur Entwicklung von Open-Source-Software beitragen, ausmachen. Aus ihren Budgets wird die Arbeit der anderen hier aufgeführten Mitwirkenden finanziert. Zusätzlich scheinen sie einen Teil ihrer Gelder ohne die üblichen Umwege direkt in die Entwicklung von Open-Source-Software zu investieren, sei es nun über die Arbeit der eigenen Mitarbeiter oder aber über ihre direkten Auftragnehmer.

Doch warum sollte ein solches Unternehmen außerhalb seiner Kernkompetenzen überhaupt einen Beitrag leisten? Da man den Ausdruck Kernkompetenzen eher auf

Individuen als auf Unternehmen bezieht, sollte man nicht darüber nachdenken, ob ein Open-Source-Projekt im Mittelpunkt der eigenen Geschäfte steht, sondern eher darüber, ob die optimale Wirtschaftlichkeit des Unternehmens dadurch erreicht wird, dass die Beteiligung am Open-Source-Projekt zur Hauptaufgabe einiger Mitarbeiter erhoben wird. Die entscheidenden Vorteile sind in diesem Fall die Kosten- und Risikosenkung und gleichzeitig die Verbesserung der Kontrolle, den man über die eigene Software hat. Sie sollten sich deshalb einmal vor Augen führen, in welchem Umfang Software Ihre Geschäftsabläufe kontrolliert. Haben Sie Kontrolle über Ihre Software?

8.5 Dienstleistungsunternehmen

Eine Reihe von Dienstleistungsunternehmen entwickelt Lösungen, bei denen verschiedene Open-Source-Programme mit anderer, direkt auf den jeweiligen Kunden zugeschnittener, Software als „Bindeglied" kombiniert werden. Andere bieten für eine bestimmte Anzahl an Open-Source-Programmen entsprechende Serviceleistungen an. Diese Unternehmensform beteiligt sich an der Entwicklung und Pflege verschiedener Open-Source-Programme, dabei jedoch selten in größerem Umfang für ein spezielles Programm. Im Allgemeinen kann man sagen, dass das Interesse des Auftraggebers das solcher Auftragnehmer häufig übersteigt, weshalb diese Dienstleister oft in die obige Kategorie Endbenutzerunternehmen und deren Auftragnehmer fallen. Einige Firmen bieten Webdienste auf Basis des Application-Service-Provider-Modells (ASP) an und bauen dabei hauptsächlich auf Open-Source-Software. Open-Source-Software hauptsächlich deswegen, weil es in vielen Open-Source-Lizenzen (insbesondere der derzeitigen Fassung der *GPL*) ein Schlupfloch gibt, das die Differenzierung solcher Unternehmen schützt. Die aus der Lizenz resultierende Pflicht, den Quellcode einer Software offenzulegen, wird erst durch die Weitergabe der Software begründet – die ASPs betreiben die Software für den Kunden jedoch lediglich und müssen sie nicht direkt zur Verfügung stellen. Da diese Lücke häufig als Makel der *GPL* angesehen wird, ist es möglich, dass sie in der nächsten Version dieser Lizenz nicht mehr existiert.

8.6 Staatliche Institutionen und Verwaltungen

Die Art und Weise, wie Open Source von staatlichen Institutionen und Verwaltungen genutzt wird, unterscheidet sich kaum von der eines Wirtschaftsunternehmens. Anders als bei einem Unternehmen wird von der Verwaltung jedoch erwartet, dass sie sich für das Wohl ihrer Bürger einsetzt, weshalb man normalerweise nicht davon ausgeht, dass die Verwaltung so etwas wie ein „Firmenkonto", auf dem Gewinne verbucht werden, besitzt. Sie stellt also eher Dienstleistungen zur Verfügung, welche die Wirtschaft ankurbeln und gesellschaftliche Aktivitäten unterstützen.

Ein Auftrag aus der Verwaltung sollte einem einzelnen Anbieter, abgesehen von den direkten Einnahmen für Produkte und Dienstleistungen, die aus dem Auftrag selbst re-

sultieren, keinesfalls einen wirtschaftlichen Vorteil verschaffen. Die Verwaltung sollte sich zur Vermeidung von *lock-in* und den damit verbundenen Wechselkosten (*switching costs*) nie an einen einzelnen Händler binden. Es gilt deshalb als schlechte Politik, wenn die Anbieter und Bürger auf ein einzelnes Produkt festlegt werden, nur um mit der Verwaltung in Kontakt zu treten, denn das würde dem Händler einen unangemessenen Vorteil verschaffen.

Jeder Händler kann im Rahmen der entsprechenden Lizenz von Open Source Gebrauch machen. So kann garantiert werden, dass die Schnittstelle zu den Einrichtungen der Verwaltung offengelegt wird und von allen gleichermaßen genutzt werden kann. Im Ergebnis kann mittels der Verwendung von Open Source ein E-Government–Modell zur Kommunikation zwischen Verwaltung und Bürgern, zwischen Verwaltung und Unternehmen und zwischen Verwaltungsapparaten untereinander ermöglicht werden.

Üblicherweise werden von der Verwaltung dem Gemeinwohl zuträgliche Aufgaben erfüllt. Die Open-Source-Entwicklungen können dann als Teil derselben durchgeführt werden. Dies erfolgt im Allgemeinen durch die Vergabe von Forschungsgeldern.

8.7 Akademiker und Wissenschaftler

Akademische Forschungsprojekte haben von jeher einen erheblichen Beitrag zur Entwicklung von Open Source geleistet, dies trifft insbesondere auf die Arbeit im Hauptstudium – eher noch als im Grundstudium – zu. Grundstudiumskurse geben den Studenten nicht genügend Zeit, sich in ein Open-Source-Projekt einzuarbeiten und selbst einen Beitrag zu leisten. Im Gegensatz dazu laufen Forschungsprojekte im Hauptstudium meist über mehrere Jahre. Ein großer Teil der Open-Source-Software wurde vom BSD-Projekt (*Berkeley System Distribution*) entwickelt, das vom *U.S. Department of Defense* finanziert wurde. Auch andere Forschungsprojekte von Studenten entwickeln Tag für Tag immer neue Software.

Es sind energische Open-Source-Communitys, die sich an solchen wissenschaftlichen Forschungen beteiligen. In der Wissenschaft gilt die Maxime *publish or perish*[15] und Open Source wird dieser Maxime gerecht. Um als stichhaltig zu gelten, muss wissenschaftliche Forschungsarbeit replizierbar sein. Wenn ein Experiment, wenn es von einem anderen Wissenschaftler durchgeführt wird, nicht die gleichen Ergebnisse erzielt, dann kann dies ein Hinweis darauf sein, dass die Forschungsarbeit fehlerbehaftet ist. Dieser Tage bildet Software einen großen Bestandteil vieler Experimente, die menschliche Sprache zur Beschreibung eines Experiments vermag jedoch nicht jedes Detail seines Ablaufs zu vermitteln. Wenn Wissenschaftler ihren Quellcode anderen

15 *Publish or perish* (dt. „veröffentliche oder gehe unter") ist eine im Wissenschaftsbetrieb, insbesondere an Universitäten, gängige Redewendung, mit der zum Ausdruck gebracht werden soll, dass Forscher einem starken informellen Druck ausgesetzt sind, ihre Ergebnisse möglichst zahlreich und in möglichst angesehenen Verlagen oder Fachzeitschriften zu veröffentlichen, um ihr wissenschaftliches Renommee zu steigern.

zugänglich machen, dann können Außenstehende ihre Software auf mögliche Fehler hin überprüfen und die Experimente leichter nachvollziehen.

Die Anzahl der Wissenschaftler, die auf einem bestimmten Forschungsgebiet tätig sind, ist relativ klein. Das Modell der Softwareentwicklung zu Vertriebszwecken wäre ungeeignet für die Entwicklung derart spezieller Software. Open-Source-Software ist insofern der beste Weg für Wissenschaftler, die Kosten für die Entwicklung von Software zur Unterstützung ihrer Forschungsarbeiten zu verteilen.

Befürworter proprietärer Software haben versucht, den großen Anteil wissenschaftlicher Beiträge an Open Source zu verringern, indem sie Partnerschaften zwischen Forschungsprojekten an Universitäten und proprietären Softwareunternehmen schlossen. Dieser Trend erscheint insbesondere dann beunruhigend, wenn staatlich finanzierte Forschungsarbeiten in Patenten resultieren, die bei den Partnerunternehmen, den Herstellern proprietärer Software, liegen. Beunruhigend deshalb, weil es sehr wahrscheinlich ist, dass diese Patente eines Tages dem Steuerzahler gegenüber, der ihre Entstehung ursprünglich finanziert hat, rechtlich durchgesetzt werden. Generell sollte staatlich finanzierte Forschungsarbeit bis ins kleinste Detail der Öffentlichkeit nutzbar gemacht werden, denn sie hat die Arbeit ja finanziert. *DARPA* (*U.S. Department of Defense Research Grant Organization* – die US-Vergabestelle von Forschungsstipendien des Verteidigungsministeriums) und die *University of California* haben das erkannt, als sie die BSD-Lizenz auf die ersten UNIX-Erweiterungen, die an der Universität entwickelt wurden, anwendeten. Diese Lizenzierung erlaubte sowohl eine Nutzung als Open Source als auch als proprietäres Produkt.

Ein Teil der Forschungsarbeit wird von Studenten ohne ein entsprechendes Entgelt betrieben. Wenn ihre Tätigkeit allerdings vergütet wird, dann stammt das Geld für die Entwicklung der Open-Source-Software meist aus dem Budget für Forschungsstipendien. Es sind in erster Linie staatliche Organe, die solche Stipendien vergeben. Ein Teil wird aber auch durch gemeinnützige Organisationen oder von Industriepartnern finanziert.

8.8 Open-Source-Mitwirkende im Überblick

Den größten Teil zu der Entwicklung von Open Source tragen immer noch die freiwillig Mitwirkenden bei. Zwischen dem unternehmerischen Bestreben nach Differenzierung und Open Source besteht ein Konflikt, dessen ökonomische Nachteile am stärksten bei jenen Unternehmen zum Tragen kommen, die normalerweise in die Entwicklung proprietärer Software investieren würden. Unternehmen, die dieser Konflikt nicht betrifft, können effektiver zur Entwicklung von Open Source beitragen. Hardwareherstellern kommt aufgrund dessen eine bedeutende Funktion zu. Auch Endanwender-orientierte Unternehmen und deren Auftragnehmer spielen eine immer bedeutendere Rolle.

9 Wie das Produktmarketing bei Open Source funktioniert

Kritiker von Open Source führen an, dass die Entwicklung von Open Source nicht fokussiert und zielgerichtet genug ist. Für gewöhnlich obliegt die Entwicklung eines Produktes einem Unternehmen, das seine Geschäftstätigkeit, angetrieben durch einen definierten Produktmarketingprozess, auf dieses Produkt ausgerichtet hat. Einige fortgeschrittene Open-Source-Projekte betreiben Produktmarketing, das mit dem eines Unternehmens vergleichbar wäre. Aber während ein Unternehmen eine Strategie entwickelt, welche die gesamte Softwareentwicklung bestimmt, gibt es für Open Source keine überregionale Planungsinstanz, die eine umfassende Strategie festlegt, an der sich die Open-Source-Entwickler orientieren. Die Aussage ist jedoch in sich falsch, insofern, als dass Open Source nicht mit einem Unternehmen vergleichbar ist. Es beschreibt eine gesamte Industrie. Eine zentrale Planungsinstanz für eine solche Industrie würde jedoch alles andere als einen offenen Markt darstellen. Das Produktmarketing der globalen Open-Source-Community orientiert sich daher eher an den Funktionsweisen der kapitalistischen Marktwirtschaft als an der Art, wie ein einzelnes Unternehmen seine Produkte planen würde.

Die Methoden, nach denen Open Source Produkt-Marketing betreibt, können als vielfach parallele Zufallsbewegungen, die in einem darwinistischen Prozess gefiltert werden, beschrieben werden. Zunächst entwickeln viele Menschen auf der ganzen Welt die Dinge, die sie selbst benötigen und schlagen dabei Wege ein, die sie sich selbst aussuchen – ohne jegliche zentrale Koordination. Auf diese Art entstehen die unterschiedlichsten potenziellen Produkte; ein Großteil von ihnen ist dabei nur für eine Handvoll anderer interessant. Es entstehen aber auch Produkte, für die sich mehr als fünfzig andere Personen interessieren und es entstehen einige wenige, die für Millionen von Menschen interessant sind.

Fünfzig Personen, die sich für ein und dieselbe Sache interessieren, aber über die ganze Welt verteilt sind, können über das Internet ein funktionierendes Softwareentwicklungsteam bilden. Die freie Zeit von fünfzig Personen, die beruflich anderweitig beschäftigt sind, erweist sich als ausreichend für die Entwicklung großer und komplexer Softwareprodukte. Selbstverständlich wächst das Entwicklerteam in der Anzahl der Beteiligten, während das Produkt reift und mehr Menschen sich dafür interessieren. So kann man unter Ausnutzung der exzellenten Zusammenarbeit, die im Rahmen von Open-Source-Projekten möglich ist, Projekte ins Leben rufen und erfolgreich durchführen, die sonst die Möglichkeiten der beteiligten Parteien übersteigen würden.

Aber wie ist es möglich, dass fünfzig Personen, die einander noch nie getroffen haben, zusammenarbeiten und ein funktionierendes Softwareprodukt entwickeln können? Einen Erklärungsansatz für die gute Zusammenarbeit liefert die Tatsache, dass Software von Natur aus sehr modular aufgebaut ist und folglich viele Personen an unterschiedlichen Segmenten der Software fast autonom arbeiten können. Sie müssen sich letztendlich nur über die Zusammenfügung der einzelnen Teile einigen. Ein

gutes Beispiel dafür ist die Debian-GNU/Linux-Distribution. Sie beinhaltet ungefähr 16 000 Programmpakete, die von mehr als 1 000 freiwilligen Entwicklern aus vielen Ländern der Welt bereitgestellt werden. Zusammengefügt stellen die Programmpakete ein zuverlässiges und gut integriertes System dar. Ein System, das Experimente in einer Weltraumfähre in der Erdumlaufbahn überwacht hat und eine Anwendergemeinschaft besitzt, die es in den Top-Fünf aller Distributionen platziert.

Aber wie ist es möglich, Open-Source-Software zu entwickeln, wenn jedem Entwickler maximaler Freiraum zugestanden wird und es keine wirkliche Autorität gibt? Das erscheint Geschäftsleuten merkwürdig, bis sie darauf kommen, dass genau so der Kapitalismus in demokratischen Ländern funktioniert. In der Wirtschaft eines kapitalistischen Staates wollen viele Unternehmen ihre Produkte ohne die Einmischung der Regierung entwickeln. Sie alle stehen zueinander im Wettbewerb und so entstehen Produkte, die sich erfolgreich am Markt etablieren können und andere, die versagen. Unternehmen arbeiten mitunter auch lose zusammen, wenn es darum geht, mit neuen Produkten neue Märkte zu erschließen. Die Rolle des Staates, der einzigen Institution, welche die Macht hätte, die Wirtschaft zu steuern, beschränkt sich im Allgemeinen darauf, der Wirtschaft Kapital zuzuführen bzw. zu entziehen, Steueranreize und Subventionsprogramme zu initiieren und die Einhaltung der Gesetze zu überwachen, die einem gerechten Marktgeschehen dienen sollen.

Die meisten Menschen akzeptieren die Tatsache, dass es so etwas wie eine „sichere Sache" im Spiel oder an den Aktienmärkten nicht gibt, und doch erwarten sie von Marketingabteilungen zuverlässige Prognosen, die der Entwicklung neuer Produkte dienlich sein sollen. Marketingabteilungen haben keine Kristallkugel. Wenn genügend viele selbstständige Entwickler vorhanden sind, können diese mit einer auf Zufallsbewegungen beruhenden Strategie erfolgreicher sein als konventionelles Produktmarketing. Im Rahmen der oben beschriebenen, vielfach parallel ablaufenden Zufallsbewegungen wird eine große Anzahl unterschiedlicher Wege beschritten, somit steigen die Chancen, einen erfolgreichen Weg einzuschlagen. Die darwinistische Filterung begründet dabei die Erkenntnis, ob ein bestimmter Weg erfolgreich ist.

Die meisten Geschäftsleute stehen fest hinter der freien Marktwirtschaft und einer Begrenzung des Einflusses, den der Staat auf das Was und das Wie der Produktion ausübt. Sie haben verstanden, dass ein freier Markt eher gute Produkte und eine gesunde Wirtschaft hervorbringen kann als eine zentrale Planwirtschaft. Es sollte sie daher nicht überraschen, dass das auf dem offenen Markt basierte Open-Source-Prinzip in der Schaffung wünschenswerter Produkte bessere Erfolge erzielen kann als zentral geleitete Marketing-Abteilungen und ihr Management.

10 Ist Open Source autark finanzierbar?

Vielen Menschen fällt es schwer zu glauben, dass Open-Source-Software sich selbst tragen kann, obwohl sie nicht in üblicher Weise als Vertriebsentwicklung entsteht. Wie

finanziert sich solcherart Software? Sie wird direkt oder indirekt aus den Budgets der Unternehmen finanziert, die sie benötigen. Diese Unternehmen wenden einen Großteil ihres Budgets für nicht-differenzierende Software auf. Sie sind deshalb gewillt, an einer Open-Source-Entwicklung dieser Software mitzuwirken, da durch die Verteilung der Kosten und Risiken auf viele Mitwirkende die diesbezüglichen Kostenzentren reduziert werden können und die Ausgaben für Software effektiver genutzt werden als bei vertriebsorientiert entwickelten Produkten. Dies sind in der Hauptsache die gleichen finanziellen Mittel, aus denen die Entwicklung proprietärer Software schöpft. Es ist wichtig, sich vor Augen zu führen, dass der Hauptfinanzier der Entwicklung eines Softwareprodukts nicht der Hersteller ist, sondern der Kunde.

10.1 Welchen Einfluss hat Open Source auf die Wirtschaft?

Wir haben den großen wirtschaftlichen Einfluss *Microsofts* wie folgt erklärt: Es ist *Microsoft* gelungen, viele Unternehmen (seine Kunden) leistungsfähiger in der Abwicklung ihrer Geschäfte zu machen. Hinzu kommt, dass diese Unternehmen ohne die Software von *Microsoft* bestimmte Geschäftsbereiche nicht mehr abdecken könnten. Gleiches trifft auch auf Open Source zu.

Auf Open Source basieren heute die meisten Webserver, der hauptsächliche E-Mail–Verkehr sowie Anwendungen in Unternehmen, Organisationen und im privaten Bereich. Somit kann Open Source auf wirtschaftliche Ergebnisse von etlichen Milliarden Dollar verweisen.

Jede technologische Verbesserung, die bewirkt, dass Geschäfte noch effizienter ablaufen können, bedeutet auch für die Wirtschaft eine Verbesserung. Open Source ermöglicht es Unternehmen, weniger für Software auszugeben und höhere Qualität sowie bessere Kontrolle über ihre Software zu erreichen. Das Geld, das an der Software eingespart werden kann, verschwindet nicht, es kann für wichtigere Dinge ausgegeben werden.

10.2 Was bedeutet dies für die Hersteller von Software?

Diese Diskussion wurde eher aus der Perspektive der Geschäftsanwender, die letztlich für die Softwareentwicklung aufkommen, als aus der Perspektive der Produzenten geführt. Aber was bedeutet das für die Hersteller von Software? Verkaufen diese fertige Software? Einige Unternehmen verkaufen eher den Prozess der Entwicklung als die Software an sich. Dies gilt im Allgemeinen für Unternehmen, die Beratungsleistungen und Lösungen anbieten. Ihre Kunden werden bereit sein, für die Entwicklung von Open-Source-Software zu zahlen.

Für ein Unternehmen, das stärker daran interessiert ist, Software zum Verkauf herzustellen, als dem Kunden Programmier- und Schulungsleistungen anzubieten, wird Open Source nur begrenzt Umsatz generieren.

Es kann durchaus sein, dass Open-Source-Software irgendwann ein Absinken der Nachfrage nach proprietärer Software verursacht. Dies bedeutet jedoch nicht, dass der Bedarf an Programmierern sinkt, denn Software wird nach wie vor ein gefragtes Gut sein. Freigewordene Programmierer proprietärer Software würden in Unternehmen abwandern, die Open-Source-Software wirtschaftlich erfolgreich produzieren können.

Es ist möglich, dass Programmierer, die in einem weniger unternehmerisch-orientierten Umfeld zu arbeiten anfangen, weniger verdienen. Allerdings sind die Gewinnüberschüsse proprietärer Softwareunternehmen bislang eher dem Management als der Belegschaft zugute gekommen. Es war bisher auch selten der Fall, dass Programmierer überraschende Gewinne aus den Belegschaftsaktien eines Unternehmens erzielt haben, das hauptsächlich Software für den Vertrieb entwickelt.

10.3 Welche Auswirkungen hat das Trittbrettfahrerproblem auf Open Source?

Das Trittbrettfahrerproblem ist in der Wirtschaft bekannt. Wie geht man mit Menschen um, die Vorteile aus einem Produkt oder einer Dienstleistung ziehen, ohne dem Anbieter eine Gegenleistung zu erbringen?

Alle Open-Source-Nutzer sind zunächst Trittbrettfahrer. Sie laden Software herunter, testen sie und setzen sie eventuell auch ein. Anfangs ziehen die meisten es nicht in Betracht, einen Beitrag zur Weiterentwicklung der Software zu leisten. Das ist erst dann der Fall, wenn sie die Software bereits gebrauchen und zusätzliche Funktionalitäten benötigen.

Wenn sie ein zusätzliches Feature benötigen, kann es sein, dass sie, anstatt einen der ursprünglichen Entwickler dafür zu bezahlen, die Software selbst um die benötigten Funktionalitäten erweitern. Kann man sie zu diesem Zeitpunkt noch als Trittbrettfahrer bezeichnen? Nein. Unternehmen, die einem Open-Source-Projekt als Entwickler beitreten, leisten einen Beitrag zu diesem Produkt und alle beteiligten Unternehmen profitieren in wirtschaftlicher Hinsicht, indem sie die Software in ihren Kostenzentren einsetzen. Die Interessen der einzelnen Entwickler von Open-Source-Software sind im Allgemeinen ähnlich, weil sie sich selbst für die Weiterentwicklung eines bestimmten Softwareprodukts entschieden haben, das für sie nutzbringend ist. Die Beiträge eines jeden Entwicklers nutzen dann auch den anderen Entwicklern.

Es gibt Entwickler, die ihre Motivation nicht aus dem Bestreben beziehen, eine Software für einen Unternehmensbereich bereitzustellen. Es sind Personen, die sich hauptsächlich aus schöpferisch-gestaltenden Beweggründen oder der wissenschaftlichen Forschung wegen beteiligen.

Freiwillig Mitwirkende empfinden die Tatsache, dass die von ihnen entwickelte Software anderen Nutzen bringt, als befriedigend, wie auch Künstler die Wertschätzung für ihre Bilder als befriedigend empfinden. Diese Freiwilligen profitieren von den Anwendern ihrer Produkte in immaterieller Hinsicht, sie stellen für sie die Motivationsquelle dar. Daher sollten auch solche Anwender nicht als Trittbrettfahrer angesehen werden.

Unternehmen, die einem bestimmten Open-Source-Produkt besondere Bedeutung beimessen, neigen dazu, Entwickler zu beschäftigen, die bereits ein gewisses Ansehen als Entwickler dieses Produkts gewonnen haben. Auf diese Weise tendieren Personen, die ohne finanzielle Interessen in ein Open-Source-Projekt eingestiegen sind, dazu, Beschäftigung in einem Unternehmen zu finden, das an dem Open-Source-Produkt ein Interesse hat. Als Resultat profitieren an der Entwicklung von Open Source Mitwirkende häufig in finanzieller Hinsicht von ihrer Teilnahme. Diese Tatsache stellt einen weiteren Grund dafür dar, dass diese Beteiligten den Anwender nicht als Trittbrettfahrer sehen sollten.

In der Wissenschaft profitiert man von einem konstanten Wissensaustausch, ähnlich dem einer Open-Source-Community, begründet dadurch, dass Wissenschaft dann am schnellsten voranschreiten kann, wenn Entdeckungen anderen Wissenschaftlern zugänglich gemacht werden und diese dann ihre Ideen beisteuern. Wissenschaftler erzielen den Lohn ihres Schaffens aus der Veröffentlichung ihrer Arbeit, weil diese ihr Ansehen bei anderen Wissenschaftlern erhöht und üblicherweise den Erfolg ihres beruflichen Werdegangs begründet. Wissenschaftler nutzen routinemäßig Open-Source-Software, um die softwarebezogenen Komponenten ihrer Arbeit zu veröffentlichen. Wissenschaftler sind ferner in ihrer Arbeit dadurch motiviert, der Gesellschaft einen Nutzen zu bringen und hegen ein Interesse daran, dass andere Menschen vom Gebrauch ihrer, durch wissenschaftliche Forschung entwickelten, Software profitieren. Für die akademischen und wissenschaftlichen Mitwirkenden an Open Source sind die Anwender ihrer Software somit von Nutzen und sollten nicht als Trittbrettfahrer gelten.

Letztlich sind die Nutzer von Open-Source-Produkten Menschen, die wir als aktive Teilnehmer in Open-Source-Projekten gewinnen können. Vergleichbar ist dies mit dem Bemühen der Händler, die breite Masse dazu zu bewegen, ihre Produkte zu kaufen. Bei einigen Anwendern geht unser Konzept stets auf.

An dieser Stelle stellt sich die Frage, ob dem Trittbrettfahrer-Problem in Bezug auf Software die gleiche Bedeutung zukommt wie für andere Arten von Produkten und ob es für Open Source überhaupt eine Rolle spielt. Jemand, der schwarz mit dem Bus fährt und die begrenzte Ressource Sitzplatz darin beansprucht, nimmt unter Umständen einem zahlenden Fahrgast die Möglichkeit, im Bus mitzufahren. Jemand, der sich illegal eine Kopie von *Microsoft Windows* hergestellt hat, beeinflusst unter Umständen die Nachfrage des Marktes, an dem Windows-Kopien gegen Bezahlung gehandelt werden. Er nutzt aber kein begrenztes Gut und schließt damit andere Windows-Nutzern aus. Ein Trittbrettfahrer, der Open Source benutzt, hat keinen negativen Einfluss auf irgendeinen Markt und verbraucht auch keine beschränkt verfügbare Ressource.

All dies lässt mich schlussfolgern, dass das Trittbrettfahrer-Problem der Open–Source-Entwicklung keinen wesentlichen Schaden zufügt.

10.4 Wenn das Open-Source-Prinzip funktioniert, warum baut sich dann nicht jeder sein eigenes Auto?

Das Open-Source-Modell funktioniert gut für Produkte, deren Gestaltung hauptursächlich für ihren Wert ist. Es ist bis heute erfolgreich eingesetzt worden, um Software zu entwickeln, eine Enzyklopädie zu erstellen sowie integrierte Schaltungen zum Einsatz in programmierbaren Halbleitern[16] zu entwerfen.

Die meisten Dinge des Lebens sind jedoch keine Software. Es kostet kaum etwas, eine Software zu kopieren. Ein ordentliches Brot herzustellen, kostet zumindest ein Pfund Mehl. Jemand muss den Weizen anbauen, diesen zu Mehl verarbeiten und all diese Anstrengungen müssen bezahlt werden.

Autos sind in ihrer Herstellung selbstverständlich viel komplexer als Brot, es wird eine große Anzahl physischer Prozesse durchlaufen und sehr teure Hilfsmittel zur Fahrzeugherstellung kommen zum Einsatz. Betrachtet man, dass zur Herstellung eines Elektromotors Erz gewonnen und zu Metall verarbeitet, Drähte gezogen, Bleche gepresst, Lager gegossen und bearbeitet und alle Teile präzise zusammengesetzt werden müssen, so ist es kein Wunder, dass für die Produktion von Autos eine ganze Industrie benötigt wird. In Kontrast dazu kann eine Einzelperson allein ein wichtiges Softwareprodukt erstellen.

Wenn es eines Tages möglich ist, dass wir komplexe dingliche Produkte erstellen können, indem wir ihr bloßes Erscheinungsbild festlegen und die Herstellung aus leicht erhältlichen Materialien und Elektrizität dann einer Maschine überlassen,[17] wird sich die Wirtschaft radikal verändern. Heute beschränkt sich unser Wirken noch auf das Produzieren einzelner Teile mit Hilfe von computergesteuerten Fräsmaschinen, einem langsamen und schmutzigen Produktionsprozess, der zudem noch immer manuelle Intervention benötigt. Eine gesunde Open-Source-Community befasst sich mittlerweile mit solchen Maschinen, und es sind auf diesem Gebiet die ersten gemeinsam entwickelten Werkstückentwürfe zu beobachten. Trotz dieser Entwicklungen muss die Forschung rund um die computergestützte Produktion noch erhebliche Fortschritte machen, damit wir eines Tages in unseren „Open-Source-Autos“ unterwegs sein können.

11 Zusammenfassung

Open Source finanziert sich selbst und funktioniert wirtschaftlich gesehen nach dem kapitalistischen Prinzip. Man muss keine Magie bemühen, um es wirtschaftswissenschaftlich zu erklären. Open Source ist ein extrem profitabler Bestandteil der freien

16 Die Entwürfe integrierter Schaltungen werden dabei in programmierbaren Halbleitern (*field programmable gate arrays* – FPGA) eingesetzt, speziellen elektronischen Bausteinen, die auf Basis eines solchen Entwurfs programmiert werden und sich daraufhin wie die entsprechende Schaltung verhalten. Hierbei handelt es sich um ein Beispiel, bei dem Hardware ähnlich wie Software verwendet wird.

17 Vergleiche die Ausführungen zum „Replikator“ unter obiger Fußnote 14.

Marktwirtschaft, da die vielen Menschen und Unternehmen, die auf das Modell aufbauen, dadurch in die Lage versetzt werden, einen ökonomisch relevanten Beitrag zu leisten. Es ist die *effizienteste* Methode der Softwareentwicklung, wenn es um die Entwicklung von nicht-differenzierenden Produkten geht, die das Unternehmen nicht von seinen Mitbewerbern abheben müssen. Nicht-differenzierende Software stellt den Löwenanteil der Software, die ein Unternehmen nutzt, und jedem Unternehmen ist anzuraten, für die Entwicklung solcher Software auf die Beteiligung an Open Source zu setzen.

Literatur

Delio, M. (2004), 'Linux: Fewer Bugs Than Rivals', *Wired.com* . http://www.wired.com/news/linux/0,1411,66022,00.html [21. Feb 2007].

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.

IDC Software Consulting (2004), 'The Linux Marketplace - Moving from Niche to Mainstream', *Prepared for Open Source Development Labs (OSDL)* . http://old.linux-foundation.org/docs/linux_market_overview.pdf [21. Feb 2007].

Kerner, S. M. (2006), 'Coverity Study Ranks LAMP Code Quality', *Internetnews.com* . http://www.internetnews.com/stats/article.php/3589361 [21. Feb 2007].

Krass, P. (2002), 'A terrible thing to waste', *CFO IT* . http://www.cfoeurope.com/displayStory.cfm/1739037 [21. Feb 2007].

Microsoft Corp. (2006), 'Form Q-10 For the Quarter Ended March 31, 2006', *Prepared for United States Securities and Exchange Commission* . http://www.microsoft.com/msft/download/FY06/MSFT_3Q2006_10Q.doc [21. Feb 2007].

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

U.S. Bureau of Labor Statistics (2006), 'Occupational Outlook Handbook (OOH), 2006-07 Edition', online handbook. http://www.bls.gov/oco/ [21. Feb 2007].

U.S. General Accountion Office (1997), 'Immature Software Acquisition Processes Increase FAA System Acquisition Risks', *Prepared for Chairman, Subcommittee on Transportation, Committee on Appropriations, House of Representatives* . http://ntl.bts.gov/lib/000/200/234/ai97047.pdf [21. Feb 2007].

U.S. Office of Technology and Electronic Commerce (2003), 'Size of the U.S. Computer Software Industry', online report. http://web.ita.doc.gov/ITI%5CitiHome.nsf/AutonomyView/87200518f179196c85256cc40077ede1 [21. Feb 2007].

Warum ist Open-Access-Entwicklung* so erfolgreich? Stigmergische Organisation und die Ökonomie der Information**

FRANCIS HEYLIGHEN

(CC-Lizenz siehe Seite 563)

Die explosionsartige Entwicklung von freien und Open-Source-Informationsgütern steht im Widerspruch zu der konventionellen Weisheit, dass Märkte und kommerzielle Organisationen für eine effiziente Güterverteilung notwendig sind. Diese Arbeit schlägt einen theoretischen Ansatz zur Erklärung dieses Phänomens vor, der Konzepte der Ökonomie und Theorien der Selbstorganisation aufgreift. Eine einmal erstellte und über das Internet zugänglich gemachte Information kann nicht mehr als knappes Gut betrachtet werden, da sie praktisch ohne Kosten vervielfältigt werden kann. Darüber hinaus profitiert der Urheber von ihrer freien Verbreitung, da dies die Qualität der Information verbessert und sein Ansehen steigert. Dies bietet Menschen bereits einen ausreichenden Anreiz, zu Open-Access-Projekten beizutragen. Im Gegensatz zu herkömmlichen Organisationen sind Open-Access-Communitys offen, dezentralisiert und selbstorganisierend. Koordination wird durch Stigmergie erreicht: Listen mit aktuell anstehenden Arbeiten, *work in progress*, führen Interessierte zu den Aufgaben, bei denen ihr Beitrag verspricht, am ertragreichsten zu sein. Dies umgeht sowohl die Notwendigkeit einer zentralisierten Planung als auch einer *unsichtbaren Hand* des Marktes.

Schlüsselwörter: Stigmergie · Informationsgüter · Soziobiologie · Community

* Gemeint ist der Herstellungsprozess von freier Software und freien Inhalten – wie in Abschnitt 1 zu lesen, unterscheidet der Autor nicht zwischen Open Access und Open Source (Anm. d. Red.).

** Aus dem Englischen von Birgit Hollenbach und Gerhard Schöppe.

1 Einleitung

In den letzten Jahren sind wir Zeugen einer überraschend schnellen und erfolgreichen Verbreitung von Free-, Libre-, Open-Access-, Open-Content- oder Open-Source–Informationsgütern geworden. In dieser Arbeit werde ich nicht auf die interessanten, doch manchmal komplizierten Unterschiede der genauen Formen der Verteilung, die mit diesen Begriffen verknüpft sind, eingehen, sondern mich auf den ihnen gemeinsamen Charakterzug konzentrieren, nämlich, dass sie nicht proprietär sind. Sie sind nicht Eigentum einer bestimmten Person oder Organisation, die das alleinige Recht zu ihrer Verbreitung hat. Sie alle sind Teil der *creative commons*[1], zu denen jeder freien Zugang hat, die jeder frei nutzen wie auch – in vielen Fällen – verändern kann. Auch werde ich die Unterschiede zwischen verschiedenen Medien oder Zielanwendungen, wie Text, Bilder, Musik und Software, ignorieren und mich stattdessen auf ihre gemeinsame Eigenschaft als pure Information konzentrieren, die ohne Begrenzung vervielfältigt werden kann. Abkürzend werde ich sie alle als „Open Access" bezeichnen. Komplexe Software-Anwendungen, Websites, Fachzeitschriften oder Magazine, Bücher, Bilder, Podcasts, Videos und sogar ganze Enzyklopädien werden von ihren Urhebern jedermann zum Nachschlagen, zum Gebrauch und sogar zum Umgestalten zur Verfügung gestellt; und das kostenlos und ohne Einschränkung. Die vielleicht bekanntesten Beispiele sind das Betriebssystem *Linux*[2] (Moody 2002), das beginnt, mit *Microsoft Windows* als Standardsystem für Heimcomputer zu konkurrieren und *Wikipedia*[3], eine internationale, im Internet verfügbare, gegenwärtig größte Enzyklopädie (Lih 2004). Weiterhin sind zu nennen *ArXiv*[4] und andere Internet-Archive, in der Tausende Forscher ihre wissenschaftlichen Ergebnisse frei zugänglich machen, bevor sie in proprietären Fachzeitschriften erscheinen (Hajjem et al. 2005), und schließlich das World Wide Web selbst, diese Sammlung von Kommunikationsprotokollen und Software-Anwendungen für die transparente Verbreitung von hypermedialen Inhalten über das Internet (Berners-Lee und Fischetti 1999).

Diese Entwicklungen revolutionieren unsere Gesellschaft. Auf der einen Seite stellen sie einen der Grundsätze der heutigen Marktökonomie in Frage, nämlich die Idee, dass geistiges Eigentum für einen Innovationsimpuls notwendig sei. Auf der anderen Seite eröffnen sie großartige Möglichkeiten:

– Software, technisches Know-how, wissenschaftliche Kenntnisse und Allgemeinbildung werden frei verfügbar – für die Länder und Menschen, die dies am meisten benötigen, es sich jedoch am wenigsten leisten können, dafür zu bezahlen.

1 *Creative commons*, aus dem englischen Originaltext so übernommen, ist eine gemeinnützige Gesellschaft, die verschiedene Standard-Lizenzverträge im Internet veröffentlicht; der Begriff lässt sich aber auch als *kreatives Gemeinwesen* übersetzen (Anm. d. Übers.).

2 http://www.linux.org

3 http://wikipedia.org

4 http://www.arxiv.org

– Die Leute werden befähigt und angeregt, intellektuell kreativ zu werden und damit anderen zu helfen.
– Es wird die Gefahr wirtschaftlicher Monopolstellungen im Bereich der Softwarestandards oder Nachrichtenverbreitung verringert.
– Erstellen wie auch Verteilen von Informationsmaterial – wann und wo es auch immer benötigt wird – erfolgt sehr viel schneller und in größerem Umfang als vorher.

Diese klar aus dem Internet resultierende Entwicklung blieb in der jüngsten Vergangenheit der Dotcom-Blase weitgehend unbeachtet, da sich die Gelehrten eher auf die vielversprechenden Möglichkeiten zur kommerziellen Ausbeutung des Netzes konzentrierten (Howcroft 2001). Mit dem Aufkommen des Webs und seinen multimedialen Fähigkeiten, Informationen zu liefern fragten sich die meisten Experten, wie große Unternehmen wie *ABC* oder *Time Warner* genug *Content* produzieren könnten, um die enorme Nachfrage nach Informationen zu befriedigen, die das Internet mit sich bringen würde. Gegenwärtig sind es die Webnutzer selbst, die diesen *Inhalt* produzieren, mit Millionen von Blogs und Wikis, Nachrichten, Meinungsforen, Unterhaltungs- und Informationsbeiträgen und dies alles zu einem Bruchteil der Kosten, welche die Unternehmen als Investition geplant hatten.

Die Entwicklung kam weniger überraschend für die Pioniere des World Wide Web (zu denen ich mich selbst zähle, da ich die komplexe Website *Principia Cybernetica*[5] bereits 1993 erstellt habe; vergleiche dazu auch Heylighen 1994). Bevor ein wirtschaftliches Interesse am Internet einsetzte, herrschte unter den Beteiligten eine Kultur der Freiheit, der Zusammenarbeit und des Teilens – nicht eine der Konkurrenz und des Ausschlusses vor. Die ersten Internetnutzer waren größtenteils Forscher, die es als selbstverständlich erachteten, die Ergebnisse ihrer Arbeit öffentlich zugänglich zu machen, ohne hierfür Geld zu verlangen. Hiermit folgten sie einfach jener grundlegenden Haltung, welche die Wissenschaft zum Antrieb von sozialer, technologischer und ökonomischer Weiterentwicklung macht: Veröffentliche deine Ergebnisse und Ideen in so weitem Rahmen als irgend möglich, so dass andere sie verwenden, kritisieren und verbessern können. Vor 1996 war praktisch jede Information und Software im Netz kostenfrei zugänglich, was ihre Erfinder jedoch nicht daran zu hindern schien, kreativ zu sein oder zusätzliches Material und verbesserte Versionen mit halsbrecherischer Geschwindigkeit zu veröffentlichen.

Dieser freie Geist wurde durch den Dotcom-Boom („com" steht für *commercial*) in den Hintergrund gedrängt, als sich Unternehmen *en masse* auf das Internet stürzten, in der Hoffnung, mittels Werbung, Verkauf oder durch geistiges Eigentum Gewinne erzielen zu können (Howcroft 2001). Das Platzen der Spekulationsblase im Jahr 2001 machte deutlich, dass das große Geld im Internet nicht so leicht zu machen war, wie viele Menschen geglaubt hatten. Zum Teil lag dies einfach daran, dass bereits sehr viel

5 http://pespmc1.vub.ac.be

kostenlos im Netz zur Verfügung stand – warum sollte man also umständlich die teure Encyclopedia-Britannica-Website abonnieren, wenn man anderswo nahezu identische Informationen kostenlos erhalten konnte? Dies bedeutet natürlich nicht, dass das Internet nicht auch für kommerzielle Transaktionen genutzt werden kann, wie es beispielsweise die äußerst erfolgreichen Unternehmen *Amazon* und *eBay* zeigen. Diese fungieren jedoch im Grunde genommen eher als Stütze des Verkaufs herkömmlicher materieller Gebrauchsgüter denn als Verkäufer reiner Informationsgüter.

Die Entwicklung lässt vermuten, dass etwas Besonderes an der Verbreitung von Informationen durch das Internet ist, das im Widerspruch zu konventionellen Aussagen über Ökonomie steht. In dieser Veröffentlichung wird versucht, die tieferliegenden Mechanismen, die dem Erfolg von Open-Access-Entwicklung und -Verbreitung zugrunde liegen, aus der Perspektive komplexer, sich entwickelnder Systeme genauer zu untersuchen. Zunächst soll jedoch ausführlicher darauf eingegangen werden, warum das Phänomen an sich für viele so erstaunlich war.

2 Wirtschaftstheorie und Open-Access-Entwicklung

Nach dem klassischen Wirtschaftsmodell sind Menschen eigennützig und würden keinen Finger rühren, um anderen zu helfen – etwa Informationsgüter zur Verfügung stellen – ohne eine Gegenleistung zu erhalten. Die traditionelle Ökonomie beruht auf der Annahme, dass Eigentumsrechte notwendig sind, um einen Anreiz zur Produktion zu schaffen. Nur wenn man die Produktion voll unter Kontrolle hat, kann man auch eine Gegenleistung von denjenigen verlangen, die das Produkt nutzen wollen. Zudem geht das freie Marktmodell davon aus, dass Wettbewerb für die Produktionsoptimierung notwendig ist: Wenn die Menschen einem die Produkte nicht mehr abkaufen, weil sie die der Konkurrenten bevorzugen, wird man gezwungen sein, das eigene Produkt entweder zu verbessern oder den Preis zu senken. Wenn alle größeren Produzenten zusammenarbeiten wie in einem Kartell, dann wird hiermit der Wettbewerb unterlaufen und die Preise können ohne eine entsprechende Qualitätsverbesserung in die Höhe steigen.

Auf den ersten Blick verstößt die Open-Access-Community gegen all diese ökonomischen Prinzipien (vgl. Lerner und Tirole 2002): Menschen produzieren Informationen oder Software kostenlos, erlauben es anderen, diese nach Belieben zu verwenden und arbeiten dabei gemeinsam an größeren Unternehmungen, wie z. B. *Linux* oder *Wikipedia*. Dennoch hat diese Community in kürzerer Zeit bei niedrigeren Kosten qualitativ bessere Produkte hervorgebracht als spezialisierte Privatunternehmen, die seit Jahrzehnten im Geschäft sind.

Noch ungewöhnlicher wird es, wenn Aspekte der Organisation und Kontrolle mit erwogen werden. Laut gängiger Theorie ist der größte Vorteil der freien Marktwirtschaft gegenüber der planorientierten Wirtschaft die Koordination. Eine zentrale Planungsinstitution könnte niemals alle notwendigen Informationen sammeln und

verarbeiten, um zu entscheiden, was wann und wo am besten herzustellen sei. Freie Marktwirtschaft hingegen funktioniert wie von einer *unsichtbaren Hand* gesteuert: Die Zuordnung der entsprechenden Menge an Rohstoffen für die Produktion jeder benötigten Ware regelt sich automatisch (von Hayek 1945). Dies geschieht durch das Gesetz von *Angebot und Nachfrage* sowie das Preissystem: Wo immer die Nachfrage nach einer Ware größer ist als das Angebot, wird der Preis höher, und die Anbieter werden somit dazu motiviert, die Produktion des Artikels zu steigern. Auf diese Weise werden Angebot und Nachfrage wie von selbst durch einen negativen Rückkoppelungsmechanismus ohne die Notwendigkeit umständlicher Planung aufeinander abgeglichen.

Die *Institutionenökonomie* fügt den Kräften des Marktes allerdings eine weitere Qualität hinzu: Ihre Hauptannahme ist, dass konkurrierende Marktteilnehmer beginnen werden, formale Kooperationen einzugehen und so Organisationen oder Firmen gründen, um Transaktionskosten zu senken (Williamson und Masten 1995). Transaktionskosten entstehen aus der Notwendigkeit, jeglichen Austausch von Gütern, Dienstleistungen oder Finanzen über Verhandlungen und Abmachungen regeln zu müssen. Diese Verhandlungen sind zeit- und arbeitsaufwändig, sie verfehlen dabei allerdings die Aufhebung grundsätzlicher Ungewissheiten (Ist dieses Produkt oder diese Dienstleistung zuverlässig? Gibt es Lücken im Vertrag?). Eine Firma baut nun auf eine Reihe vereinbarter Vorschriften, die das Zusammenspiel ihrer Beschäftigten steuern, um ständige Verhandlungen und Ungewissheiten zu vermeiden. Dies erfordert das Ziehen einer klaren Grenze zwischen denjenigen, die der Organisation zugehörig sind (und daher ihre Vorschriften befolgen sollen) und denen, die nicht dazugehören. Damit verringert sich auch die Möglichkeit, dass vertrauliche Informationen aus dieser Gruppe nach außen dringen, wo sie von Konkurrenten ausgenutzt werden könnten.

Zwar mögen Vorschriften verlässlicher sein als ein freier Markt, doch sind sie weniger flexibel, da sie sich nicht automatisch an neue Bedingungen anpassen können. So benötigt die Organisation eine intelligente Führung, um die Tätigkeiten der Mitarbeiter zu koordinieren und sie dorthin zu leiten, wo die wichtigsten Aufgaben vorliegen. Dies wird normalerweise durch eine hierarchische Struktur erreicht, mit einem CEO oder einem Führungsstab an der Spitze, der die Arbeit plant und kontrolliert sowie Anweisungen an die unteren Ebenen erteilt. Ein derartiges Kontrollsystem muss außerdem sicherstellen, dass die Angestellten nach den Regeln spielen und nicht etwa eigennützig handeln, ohne die gebührende Leistung zu erbringen. Anders ausgedrückt muss das Management der allgegenwärtigen Gefahr, die von *Trittbrettfahrern* ausgeht, im Bedarfsfall durch wirkungsvolle Sanktionen entgegenwirken.

Das Paradoxe an der Open-Access-Community ist, dass sie die meisten dieser Organisationsprinzipien zu ignorieren scheint. Im Allgemeinen kann jeder zu jedem beliebigen Zeitpunkt einer bestimmten Community beitreten oder sie wieder verlassen, es gibt keine offiziellen Mitglieder oder Angestellte, da die Beitragenden dazu neigen, sich in unterschiedlichem Maße zu beteiligen. Zudem ist die Community üblicherwei-

se dezentralisiert, ohne eine offizielle, hierarchische Struktur oder Bestrafungen für Trittbrettfahrer. Raymond (1999) bezeichnet dieses lose, sich selbst organisierende Kooperationsmodell als *Basar*, im Gegensatz zu dem Modell der *Kathedrale*, welches sich durch geschlossene, zentralisierte, hierarchische Organisationen auszeichnet. Auch kann sich die dezentralisierte Koordination nicht auf Marktmechanismen verlassen, da keine Produktpreise existieren, die anzeigen, wo die Nachfrage am höchsten ist.

Zusammengefasst läuft die Open-Access-Entwicklung nicht nur den üblichen Weisheiten der Geschäftswelt zuwider, sondern auch einigen grundlegenden Annahmen der Wirtschaftstheorie. Dies bedeutet, dass hier eine alternative Theorie zu entwickeln ist, um erklären zu können, wie Open Access funktioniert.

3 Anreize für information sharing[6]

Die erste grundlegende Eigenschaft der Information ist in ihrer nichtmateriellen Natur angelegt. Information ist nicht dem physikalischen Gesetz des Erhalts von Masse und Energie unterworfen und kann daher auch keiner wirtschaftlichen Beschränkung durch Knappheit unterliegen. Wenn man erst einmal ein Stück Information, beispielsweise in Form eines Computerprogramms, erhalten hat, kann man dieses uneingeschränkt und nahezu ohne weitere Kosten vervielfältigen und verbreiten. Das Verteilen dieser Kopien an andere raubt die Information jedoch nicht dem ursprünglichen Besitzer. Die Tatsache, dass man eine bestimmte Information verwendet, schließt die gleichzeitige Verwendung der gleichen Information durch andere in keiner Weise aus. Diese Eigenschaft wird in der Ökonomie als *non-rivalry*[7] (z. B. Martens 2004) bezeichnet. Die grundlegende Annahme der wirtschaftlichen Knappheit wird hier ins Gegenteil verkehrt. Wie mit nichtrivalisierenden Gütern umzugehen ist, wird in Wirtschaftstheorien jedoch kaum erörtert (De Long und Froomkin 1998).

Eine weitere wichtige Eigenschaft von Informationen ist ihre nur *unvollständige Ausschlussmöglichkeit* gegenüber anderen. So ist es zwar möglich, andere von der Nutzung eines Informationsgutes auszuschließen – etwa durch Copyright-Rechte, Patentierung oder Kopierschutz. Doch werden diese Verbote durch die einfach zu bewerkstelligende Vervielfältigung von Informationen zunehmend schwerer umsetzbar. Dies stellt für die herkömmliche Ökonomie ein Problem dar, da davon ausgegangen wird, dass jeder Produzent einer Ware, gleichgültig ob diese Ware rivalisierend ist oder nicht, einen Anreiz für die weitere Produktion der Ware benötigt und Maßnahmen erforderlich sind, um Konsumenten anzuhalten, für das Produkt zu zahlen (Martens 2004; De Long und Froomkin 1998).

In Open-Access-Communitys sehen wir jedoch, dass ganz andere Anreize vorherrschen. Zuerst einmal erklärt die Non-rivalry-Eigenschaft von Informationen, warum

6 Zur Erläuterung des Begriffs *sharing* und zur Problematik seiner Übersetzung möchte die Redaktion auf den Artikel von Bernd Lutterbeck auf Seite 461 in diesem Buch hinweisen.

7 Deutsch: nichtrivalisierend (Anm. d. Red.).

Open-Access-Entwicklung auch mit wenigen Anreizen funktioniert. Angenommen, man hat die notwendigen Mittel (Hardware, Software, Expertise, Zeit etc.) zur Verfügung, so kann man ein Informationsgut zur eigenen Verwendung erstellen (ein Programm, eine Bibliografie, ein Gedicht, ein Foto von seinem Hund usw.) und dieses Gut dann auch anderen ohne großen zusätzlichen Aufwand zur Verfügung stellen. Auf diesem Weg kann mit geringem Einsatz der eigenen Zeit im Rahmen eines Hobbys, einer zufälligen Entdeckung oder eines schnell getippten *hacks*[8] trotzdem ein Informationsgut entstehen, dass tausend anderen von Nutzen ist (vgl. Ghosh 1998).

Auch wenn die direkten Auswirkungen voraussichtlich eher gering sind, kann der Nutzen für andere auch dem Urheber selbst indirekt zugute kommen. Selbst wenn die Benutzer in der Umgebung durch die Verwendung des Beitrags nur ein bisschen effizienter oder produktiver arbeiten können oder einfach glücklicher werden, dann bedeutet dies schon eine Verbesserung des eigenen Lebens in der Gemeinschaft, auch wenn niemand weiß, wer der Urheber ist. Die Vorstellung, etwas zur Gemeinschaft beizutragen appelliert an den Instinkt von Altruismus oder *Wohltätigkeit*, der auch Anlass ist Geld, für karitative Zwecke zu spenden oder ehrenamtlich tätig zu werden. Dies ist ein erster Anreiz für die gemeinschaftliche Nutzung von Informationsgütern (Weber 2004). Obwohl in der Soziobiologie und der evolutionären Psychologie (Buss 1995) nachgewiesen werden konnte, dass dem Altruismus ein klares genetisches Motiv zugrunde liegt, scheint Altruismus allein keine ausreichend starke Eigenschaft zu sein, um ein komplexes Wirtschaftssystem zu tragen, wie das Trittbrettfahrerproblem (vgl. Heylighen 2007) und das Versagen des Kommunismus zeigen.

Zunächst sollten wir festhalten, dass das klassische Trittbrettfahrerproblem nicht für Non-rivalry-Produkte gilt. Ein Trittbrettfahrer ist jemand, der von dem Einsatz anderer profitiert, ohne eine entsprechende Gegenleistung zu erbringen. Die meisten Nutzer von Open-Access-Informationen fallen unter diese Kategorie: Sie verwenden von anderen erstellte Produkte, tragen selbst aber nichts oder wenig zu ihnen bei. Dieses offensichtliche „Parasitentum" oder der fehlende Grundsatz der Gegenseitigkeit ist nicht wirklich ein Problem, denn die Urheber profitieren nicht etwa weniger von ihren Produkten, nur weil andere ebenfalls Zugriff darauf haben. Dies erklärt bereits, warum in Open-Access-Communitys die üblichen Überwachungs- und Bestrafungsmechanismen fehlen, die in anderen sozialen Gruppen im Laufe der Geschichte entwickelt wurden, um Profiteure abzuschrecken. Im Folgenden werden wir außerdem zeigen, dass die Urheber von ihrem Produkt mehr als die bloßen *Konsumenten* profitieren und sich so einen wesentlichen Wettbewerbsvorteil gegenüber Trittbrettfahrern sichern.

Im Folgenden sollen die mehr eigennützigen Motive betrachtet werden, die dazu führen, dass man eine Arbeit Open Access zur Verfügung stellt. Der minimale Aufwand, den es kostet, seine Ergebnisse mit anderen zu teilen, wird mehr als genug durch die Tatsache entschädigt, dass andere Verbesserungen am Produkt vorschlagen

8 Ein kleines effektives Computerprogramm oder eine Codeverbesserung.

oder durchführen die sich für den Urheber lohnen. Beispielsweise könnte das Interesse an einer bestimmten historischen Persönlichkeit oder einem geografischen Ort dazu führen, dass man sein Wissen in einem Artikel in der *Wikipedia* veröffentlicht. Es ist wahrscheinlich, dass dieser Artikel andere dazu anregt weitere, dem Autor bisher unbekannte Details hinzuzufügen und ihm so weiterzuhelfen. Die gewollte und öffentlich zugängliche Präsentation von Informationen führt zusätzlich zu Feedback (wobei es sich auch um eine nette Bemerkung wie „gute Arbeit" handeln kann oder darum, dass man darauf aufmerksam gemacht wird, dass ein Programm nicht ganz so genutzt wird, wie man erwartet hat). Feedback („Rückmeldung") ist die Grundlage des Lernens. Selbst wenn das eigentliche Produkt durch die Veröffentlichung nicht verbessert wird, so wächst doch vielleicht die eigene kreative Kompetenz.

Wenn die Möglichkeit, ein besseres Produkt zu erhalten und sich selbst zu verbessern nicht Anreiz genug bietet, gibt es eine noch stärkere Motivation, einen Beitrag zur Gemeinschaft zu leisten: Man wird hierdurch innerhalb der Gemeinschaft sichtbar und erntet Anerkennung für seine Expertise, seine Arbeit und seinen Altruismus. Psychologen nehmen an, dass der Gewinn an Achtung, Anerkennung oder hohem Ansehen innerhalb der eigenen Gemeinschaft ein elementares menschliches Streben ist (Maslow 1970). Aus der Sicht der evolutionären Psychologie (Buss 1995) ist dieses Streben wahrscheinlich sogar elementarer als das Streben nach Reichtum, da auch (prä-)historisch gesehen die Person mit dem höchsten Status in der Gruppe sowieso den besten Zugang zu materiellen Ressourcen erhält – zusätzlich zu Partnern, Freunden, Unterstützung oder auch Informationen. Studien haben ergeben, dass das Erwerben eines guten Rufes tatsächlich einen entscheidenden Anreiz für viele darstellt, sich an Open-Source-Entwicklungen zu beteiligen (Lerner und Tirole 2002).

Einer der Wege, wie sich diese Entwickler ein herkömmliches Einkommen sichern können, ist die Erhebung eines Beratungshonorars für das Produkt (z. B. Software oder Blogs), das selbst jedoch kostenlos zur Verfügung gestellt wird. Ist eine komplexe Software erst einmal von vielen Nutzern angenommen worden, so könnten einige der Anwender bereit sein, für Hilfe bei speziellen Problemen zu bezahlen und sich hierfür an die einzig wirklichen Experten, nämlich die Erfinder der Software, wenden wollen. Urheber von Produkten mit literarischem oder künstlerischem Inhalt könnten ähnliche Methoden anwenden. Beispielsweise könnte eine Rockband ihre Aufnahmen kostenlos zur Verfügung stellen und wenn sie auf diese Weise ausreichend bekannt geworden ist, ihre Einnahmen durch Konzertveranstaltungen sichern. Ebenso gut könnten Schriftsteller oder Journalisten kostenlosen Zugang zu ihren Texten (z. B. Blogs) ermöglichen und Geld durch Interviews oder Lesungen für ein limitiertes Publikum verdienen.

Diese Art des Geldverdienens macht mehr Sinn, als den Zugang zu proprietären Informationen kostenpflichtig zu machen, da es sich auf das Einzige bezieht, an dem es in einer Informationsgesellschaft zunehmend mangelt: persönliche Aufmerksamkeit (Simon 1971). Wie viel Informationen auch immer von einer Person frei zugänglich

gemacht werden, ein wichtiger Teil der Fachkenntnis des Urhebers bleibt immer verborgen (Reber 1993) und daher für eine einfache Vervielfältigung unzugänglich. Solch ein intuitives Wissen kommt erst dann zum Tragen, wenn es gilt, ein ganz bestimmtes Problem zu lösen oder eine Frage zu beantworten. Da jede Person nur über eine begrenzte Zeit, Energie und Aufmerksamkeit verfügt, um sich mit einem speziellen Thema zu befassen, besitzt dieses verborgene Wissen nicht die Eigenschaften von *non-rivalry* oder uneingeschränkter Vervielfältigung, die charakteristisch sind für explizite Informationen. Das Gesetz des Marktes bestimmt daher, dass Menschen bereit sind, zu bezahlen, um die persönliche Aufmerksamkeit eines Experten zu erhalten.

Wenn man jedoch seinen Rat für gutes Geld verkaufen will, so muss man den Kunden von seinem Sachverstand überzeugen. Hierbei spielt der Ruf eine wichtige Rolle. Ein noch direkterer Weg, um Anerkennung für die eigene Expertise zu erlangen, ist, die Produkte öffentlich (und möglichst kostenlos) zugänglich zu machen, so dass die Menschen sich selbst überzeugen können, wie gut man ist. Dies wird jedoch sehr viel schwieriger, wenn man in einer Firma arbeitet, denn dort ist es für einen Außenstehenden kaum ersichtlich, wer was innerhalb dieser geschlossenen Organisation beigetragen hat (vgl. Lerner und Tirole 2002). Ein ähnliches Verfahren wird bereits in akademischen Kreisen angewandt: Die wissenschaftliche Fachkenntnis wird anhand der Quantität und Qualität der Publikationen gemessen, und das Einkommen orientiert sich stark an dieser Wahrnehmung von Fachkenntnis. Studien haben gezeigt, dass eine Open-Access-Publikation die Anzahl der Zitationen erhöht, welche das direkteste Maß für die Wahrnehmung und das Ansehen des Autors darstellt (Hajjem et al. 2005).

Was dieser Art der Entlohnung jedoch fehlt, ist ein Weg, die anfänglichen Investitionen zu bezahlen, die vor dem Erstellen von verwertbaren Informationen gemacht werden müssen. Dies beinhaltet Hardware, Software, Internetzugang und Bildung. Glücklicherweise sinken die Preise all dieser notwendigen Faktoren dank des andauernden Fortschritts in der IT. Der teuerste Teil, die allgemeine Schulbildung, wird in den meisten Ländern (nahezu) kostenlos angeboten, da die finanzielle Belastung von der gesamten Gesellschaft getragen wird, die längst verstanden hat, dass gut ausgebildete Arbeitskräfte allen zugute kommen. Weiterführende Bildung bleibt üblicherweise teuer, wenn es auch meist Stipendien für diejenigen gibt, deren Zukunft am vielversprechendsten erscheint (z. B. gemessen an den Arbeiten, die sie bereits veröffentlicht haben). Aber auch hier gilt das Versprechen, dass Fortschritte in der Informationsverarbeitung und -verbreitung und der damit einhergehenden Open-Access-Philosophie bald dazu führen werden, dass Fortbildungsmaterialien für alle Bereiche im Internet frei zur Verfügung gestellt werden können.[9]

9 Siehe z. B. http://www.wikibooks.org und http://www.globaltext.org.

4 Selbstorganisation durch Stigmergie

Um die dezentralisierte Organisation zu verstehen, die Open-Access-Entwicklungen charakterisiert, lassen sich Anregungen aus den neuesten Theorien zur Selbstorganisation (z. B. Heylighen und Gershenson 2003) und der komplexen adaptiven Systeme beziehen (z. B. Muffatto und Faldani 2003). Ein hier bedeutsames Konzept, das in kollektiven Intelligenzmodellen (Heylighen 1999) und der Simulation von Schwarmverhalten (Bonabeau et al. 1999) angewandt wird, ist das der Stigmergie (Susi und Ziemke 2001). Ein Prozess ist stigmergisch, wenn eine von einem Agenten begonnene Arbeit (im Griechischen *ergon*) ein Zeichen (*stigma*) hinterlässt, der andere Agenten dazu anregt, diese Arbeit fortzusetzen.

Dieses Konzept wurde ursprünglich entwickelt, um zu erklären, wie ein „Basar" unintelligenter und unkoordinierter Termiten den Bau ihrer aufwändigen, „kathedralenähnlichen" Termitenhügel bewerkstelligt (Grassé 1959). Die zugrundeliegende Annahme ist, dass eine Termite an einer beliebigen Stelle ein wenig Tonerde ablegt, und dass die so entstehenden Haufen andere Termiten dazu anregen, an gleicher Stelle etwas hinzuzufügen, statt den Aufbau eines eigenen Haufens zu beginnen. Auf diese Weise wachsen die Haufen in die Höhe bis sie andere, ähnlich konstruierte Säulen berühren. Die Termiten kommunizieren nicht darüber, wer wann wo was oder wie zu tun hat. Die einzig bestehende Kommunikation ist indirekt; die von einigen begonnene Arbeit liefert anderen die Information, wo sie ihren eigenen Beitrag leisten können. Auf diese Weise besteht keine Notwendigkeit für einen zentral kontrollierten Plan, ein Arbeitsschema oder organisierte Arbeitsteilung.

Obwohl Menschen gewiss wesentlich intelligenter sind als soziale Insekten und miteinander kommunizieren, funktioniert die Entwicklung von Open Access jedoch im Wesentlichen nach dem gleichen Stigmergie-Prinzip (vgl. Elliot 2006 sowie die Simulation von Robles et al. 2005). Jegliches Dokument und auch jede Softwarekomponente, die auf die Website einer Community gestellt oder dort überarbeitet wird, ist umgehend Gegenstand intensiver Begutachtung durch interessierte Mitglieder der Community. Sobald eines von ihnen eine Schwachstelle entdeckt, etwa einen Bug, einen Irrtum oder eine fehlende Funktionalität, wird das Communitymitglied versuchen, den Fehler selbst zu beheben oder ihn zumindest dem Rest der Community mitteilen, was wiederum jemand anderen zur Problemlösung anregen könnte.

Wie bei der stigmergischen Organisation von Insekten (Bonabeau et al. 1999) ist der Prozess auch hier selbstverstärkend oder autokatalytisch (Heylighen 1999; Heylighen und Gershenson 2003): Je mehr hochqualitatives Material sich auf der Website der Community befindet, desto mehr Menschen werden sich animiert fühlen, es zu begutachten – folglich stehen noch mehr Menschen zur Verfügung, um weitere Verbesserungen vorzunehmen. Open Access profitiert demnach von einem positiven Rückkoppelungsmechanismus, der erfolgreiche Projekte noch weiter ankurbelt. Dies erklärt das explosionsartige Wachstum von Systemen wie *Wikipedia* oder *Linux*. (Ein

möglicher Nachteil dieser „reich wird reicher"-Dynamik ist, dass gleich gute, wettbewerbsfähige Projekte aufgrund von Zufallsschwankungen oder Sequenzeffekten nicht die kritische Masse zum *take-off* erreichen.)

Auch wenn den meisten großangelegten Open-Access-Projekten (wie z. B. *Linux*) eine oder mehrere zentrale Persönlichkeiten (wie z. B. Linus Torvalds) vorstehen, die die allgemeine Richtung vorgeben, in die sich das Projekt bewegen soll, ist diese Kontrolle immer noch wesentlich weniger strikt als in herkömmlichen hierarchisch organisierten Strukturen. Die Arbeit erfolgt normalerweise dezentral und selbstorganisierend. Das Fehlen detaillierter Planung wird mehr als kompensiert, da Informationen über den aktuellen Projektstand vollständig und kostenlos zugänglich sind und es so jedem möglich ist, jederzeit zu irgendetwas beizutragen. Dadurch ergibt sich ein sehr viel breiteres Spektrum an Perspektiven und Erfahrungen, die angewendet werden, um das Auffinden und Lösen von Problemen zu erleichtern, was Raymond (1999) als „Linus' Gesetz" bezeichnet: „Given enough eyeballs, all bugs are shallow".[10] Da außerdem die Beitragenden ihre Aufgabenbereiche selbst wählen, werden sie im Regelfall auch interessierter, motivierter und sachkundiger in diesem Bereich sein.

Auf diese Weise profitiert Open-Access-Entwicklung von der evolutionären Dynamik der *Variation*, *Rekombination* und *Selektion* (van Wendel de Joode 2004; Muffatto und Faldani 2003). Offenheit zieht eine größere Anzahl verschiedenartiger Teilnehmer an, wodurch die Wahrscheinlichkeit einer Kreuzbefruchtung ihrer Ideen steigt, die zu neuen Kombinationen führt. Hierdurch nimmt die Variation stark zu, was für evolutionäre Neuentwicklungen notwendig ist. Darüber hinaus wird Selektion in einer großen und vielfältigen Community gefördert, da neue Ideen unter vielen verschiedenen Bedingungen ausgetestet werden und dadurch Fehler und Schwachstellen, die in einem homogeneren Umfeld nicht aufgefallen wären, systematisch behoben werden. Alles in allem führt dies zu größerer Flexibilität, Innovation und Funktionssicherheit.

Stigmergie ist jedoch mehr als nur zufällige Variation und natürliche Selektion. Die sichtbaren Spuren der durchgeführten Arbeit fungieren als Mittlersystem (Heylighen 2007), indem Informationen gespeichert und der Community (indirekt) mitgeteilt werden. Auf diese Weise koordiniert dieser Mittler weitere Tätigkeiten zu den Aufgaben, wo eine Bearbeitung am ertragreichsten sein wird. Hierfür wird ein gemeinsamer Arbeitsbereich (*shared workspace*) benötigt, der für alle Beitragenden zugänglich ist (ähnlich wie das Blackboard-System der KI). In diesem externen Datenspeicher wird vermerkt, welche Aufgaben bereits erledigt wurden und welche Probleme noch offen sind. Einen mächtigen derartigen Arbeitsspeicher bietet das Web, da es den Speicher bereitstellt und *sharing* von jeglichem Work-in-progress-Informationsgut ermöglicht.

Um besser verstehen zu können, auf welche Art und Weise Open-Access-Communitys funktionieren, müssen wir genauer zwischen direkter und indirekter Stigmergie unterscheiden. Bei direkter Stigmergie, wie am Beispiel des Termitenhügelbaus ge-

10 Zu Deutsch etwa: „Alle Bugs verlieren an Größe, wenn sie nur von genügend Augenpaaren betrachtet werden" (Anm. d. Red.).

zeigt, ist es der Stand des Arbeitsprozesses selbst, der den weiterführenden Beitrag fördert. Indirekte Stigmergie hingegen kann anhand des Beispiels von Ameisen verdeutlicht werden, die Spuren aus Pheromonen erzeugen und dadurch andere Ameisen zu Futterquellen leiten. Die Spur selbst ist nur ein „Nebenprodukt" der eigentlich durchgeführten Arbeit – der Nahrungssuche und deren Transport zum Nest. Die speziell hierfür produzierte Signalspur wird benötigt, weil die Aufgabe – die sprichwörtliche „Nadel (Futter) im Heuhaufen (Umgebung) finden" – ohne detaillierte Hinweise sonst zu komplex wäre. Indirekte Stigmergie verwendet also ein zusätzliches Hilfsmittel für die Informationsspeicherung. Trotzdem trägt die hieraus entstehende Koordination weiterhin den Stempel der dezentralisierten Selbstorganisation: Die Information ist an niemand Bestimmten direkt gerichtet und kann – aber muss nicht – von einem einzelnen Individuum in einem bestimmten Moment aufgegriffen werden.

In Open-Access-Entwicklungen kann indirekte Stigmergie in solchen Internet–Foren beobachtet werden, in denen Bugs oder Wünsche zu Programmfunktionen eingestellt werden. Die Foren selbst sind nicht Teil des sich entwickelnden Informationsguts, werden aber regelmäßig von dessen Entwicklern zu Rate gezogen und richten somit deren Aufmerksamkeit auf die Aufgaben, die für Weiterbearbeitung lohnenswert erscheinen. Das Problem solcher Hilfsmittel ist allerdings, dass die Komplexität der (Selbst-)Organisation erhöht wird, insbesondere wenn eine Menge potenziell relevanter Informationen eingestellt wird, so dass sich Prioritäten nur noch schwer setzen lassen. Auch hier können wir eine Lektion von den sozialen Insekten lernen. Die Spuren, die Ameisen durch Pheromonsignale setzen, zeigen eine effiziente Form des sich verstärkenden Lernens (Heylighen 1999): Spuren, die zu reichhaltigen Futterquellen führen, werden von vielen Ameisen genutzt und dadurch verstärkt, während Spuren zu ungenügenden oder leeren Futterstellen verblassen und letztendlich verschwinden. Da Ameisen bevorzugt der markanteren Spur folgen, sichert dies, dass die wichtigsten Aufgaben oder Angebote zuerst angegangen bzw. genutzt werden.

Auf Open-Access-Entwicklung angewandt, heißt dies, dass anpassungsfähige Mechanismen notwendig sind, um die dringlichsten Anfragen hervorzuheben. Ein Beispiel für einen solchen Mechanismus kann man bei *Wikipedia* finden: Wenn ein Wort von einem Nutzer dort als Hyperlink markiert wird, aber noch kein Artikel zu diesem Begriff existiert, wird eine leere Seite erstellt. Dies soll andere einladen, diese leere Seite nun mit Inhalt zu füllen. Es handelt sich hier um direkte Stigmergie; immer, wenn Leute nach diesem speziellen Begriff suchen, werden sie sofort zu der Aufgabe geleitet, die noch erledigt werden muss. Doch wenn mehrere tausend noch unvollständige Einträge existieren, müssen Prioritäten gesetzt werden. Statt nun ein Zentralkomitee entscheiden zu lassen, welche Einträge am dringlichsten sind, wird bei *Wikipedia* eine einfache Form der gemeinsamen Entscheidungsfindung angewandt (vgl. Heylighen 1999): Die Einträge, zu denen die meisten Hyperlinks führen, werden als erste in der automatisch erstellten Liste „Wikipedia:Most wanted articles"[11] aufgelistet.

11 In der deutschen *Wikipedia* ist diese Liste unter „Wikipedia:Fehlende Artikel" zu finden (Anm. d. Red.).

Ein solcher Mechanismus, der die kollektive Nachfrage abbildet, kann als nicht–proprietäre Entsprechung des Marktes gesehen werden. Dort ordnet der Preismechanismus sehr effizient Ressourcen zu jener Produktion von Gütern zu, nach denen die Nachfrage am größten ist, indem er die größtmögliche finanzielle Belohnung bietet. Ähnlich hierzu bietet ein Most-wanted-Auftrag die größte Aussicht auf Anerkennung für geleistete Arbeit wie auch Zufriedenheit aufgrund der altruistischen Tat. Eine solche stigmergische Priorisierung funktioniert wohl sogar noch effizienter als ein Markt, da hier keine Notwendigkeit für umständliche und oft irrationale Kauf- und Verkaufshandlungen, Tarifrunden oder Spekulationen besteht, die letztendlich den Preis einer Ware festlegen – was zu den bekannten chaotischen Preisschwankungen von Gütern an der Börse führt. Obwohl darüber hinaus der Preis als – sehr abstraktes – stigmergisches Signal ausgelegt werden könnte, ist er jedoch lediglich eine eindimensionale Größe. Dagegen können Open-Access-Aufgaben auf einer Website nach voneinander unabhängigen stigmergischen Kriterien geordnet werden, wie Dringlichkeit, Schwierigkeitsgrad, erwarteter Nutzen, notwendige Fachkenntnis etc. Auf diese Weise würde Interessierten geholfen, die am besten auf sie zugeschnittene Aufgabe zu finden.

5 Fazit

Seit dem Zerfall der Sowjetunion wird allgemein angenommen, dass Märkte, private Eigentumsrechte und kommerzielle Organisationen für die effiziente Produktion und Verbreitung von Gütern notwendig seien. Diese Ansicht wird nicht nur durch das anscheinende Scheitern des Kommunismus, sondern auch durch zwei Jahrhunderte wirtschaftswissenschaftlichen Denkens gestützt, aus denen ausgeklügelte (Denk-)Modelle hervorgingen, die vorgeben, aufzuzeigen, dass der Markt die optimale Möglichkeit für die beste Allokation von Rohstoffen sei. In den letzten Jahren jedoch tauchte die gemeinschaftliche Entwicklung von Open-Access-Informationsgütern im Web als augenfällige Ausnahme zu dieser konventionellen Lehre auf. In der hier vorliegenden Arbeit wurden Erklärungsansätze für dieses Phänomen vorgeschlagen.

Ich habe zunächst aufgeführt, dass sich die wirtschaftlichen Grundbegriffe Wettbewerb und Ausschlussprinzip der Güternutzung nicht auf die gemeinsame Informationsnutzung im Internet anwenden lassen. Eine einmal erstellte Information kann nicht mehr als knappes Gut betrachtet werden, daher gibt es grundsätzlich auch keinen Anlass, den Zugang zu ihr zu beschränken. Im Gegenteil ist es eher wahrscheinlich, dass der Urheber von seiner frei verbreiteten Information profitiert, da hierdurch die Qualität der Informationen verbessert werden kann und die Fachkompetenz und das Ansehen des Urhebers steigen. Darüber hinaus kommt Open Access eindeutig allen anderen zugute, insbesondere denjenigen, die andernfalls zu arm wären, um sich den Zugang zu Informationen leisten zu können.

Weiter benutzte ich das Paradigma der Selbstorganisation durch Stigmergie, um zu erklären, wie Open-Access-Entwicklung effizient koordiniert werden kann. Dank

der Websites, die Work-in-progress-Projekte auflisten, werden Interessierte, die sich an der gemeinschaftlichen Entwicklung eines Informationsguts beteiligen wollen, erfolgreich zu den Aufgaben geleitet, bei welchen ihr Beitrag höchstwahrscheinlich am ertragreichsten ist. Hierbei ist sowohl die Notwendigkeit einer zentralisierten Planung als auch einer *unsichtbaren Hand* des Marktes, die Angebot und Nachfrage aufeinander abstimmt, überflüssig.

Diese hier beschriebenen Innovationen scheinen derart fundamental, dass sie unser sozio-ökonomisches System revolutionieren können (vgl. Weber 2004) und bieten Anlass zu der Hoffnung, in der Zukunft etwa die Entwicklung von Bildung, Demokratisierung und Wirtschaft anregen zu können. Während Open-Access-Verteilung auf materielle Güter nicht anwendbar ist, wird der Anteil der Materialkosten an den Gesamtkosten jeder Ware oder Dienstleistung immer geringer in einer Gesellschaft, die immer stärker von Informationen abhängig ist.

Daher wird es theoretisch vorstellbar, dass der größte Teil an wirtschaftlicher Wertschöpfung letztendlich in einem Open-Access-System entsteht. Um ein solches Szenario weniger spekulativ erscheinen zu lassen, wird man zunächst das komplexe Thema der Informationserstellung dort zu untersuchen haben, wo ein beträchtliches Maß an materieller Investition erforderlich ist, wie etwa in der pharmazeutischen Forschung mit ihrer kostenintensiven Ausstattung. Dort also, wo Patente und andere Dokumente mit „aussperrenden" Funktionen überhand nehmen. Dieses Vorhaben wirkt weit weniger entmutigend, wenn man sich bewusst wird, dass diese Art von Forschung meist aus öffentlichen Mitteln finanziert wird und somit normalerweise ein Open-Access-Gut ist.

Um mit dem bestehenden marktorientierten System tatsächlich konkurrieren zu können, wird die immer noch junge Open-Access-Gemeinde weiter aus ihren Erfahrungen lernen müssen, also an ihren verbleibenden Schwächen arbeiten und weiter auf ihre Stärken bauen. Hierfür ist vor allem die Entwicklung besserer Standards und Regeln sowie leistungsfähigerer Softwarelösungen notwendig, um das Stigmergieprinzip gebührend einzubinden und den Akteuren ihre Anerkennung und Resonanz zu bieten, welche nach der bisherigen Analyse die wesentlichen Antriebsfaktoren für den Erfolg von Open Access sind.

Beispielsweise ist es zurzeit im Wikipedia-System – das ansonsten alle Änderungen an allen Dokumenten von allen Nutzern sehr detailliert verfolgt – nicht möglich, einen Überblick darüber zu erhalten, wie viel ein bestimmter Nutzer bisher schon zum System beigetragen hat. Mit dem Versionskontrollsystem von *Wikipedia* sollte es jedoch möglich sein, zu ermitteln, wie viel der eingegebenen Textbeiträge eines Nutzers im aktuellen Stand der Enzyklopädie bestehen bleiben. Dies wäre ein verwendbares Maß für die Quantität und Qualität der Beiträge und würde einem Bewertungssystem für Fachkompetenz und Aktivität des Autors entsprechen. Auch könnte die Aufgabengliederung und -priorisierung durch Verwendung weiterentwickelter Algorithmen (aufbauend etwa auf Googles *PageRank* oder der Hebbschen Lernregel) ermöglicht

werden (Heylighen 1999). Der Einsatz intelligenter Methoden für die Koordination der Verbreitung von Informationsgütern könnte das World Wide Web von einem bloßen gemeinschaftlich genutzten Datenspeicher oder einem geteilten Arbeitsraum in ein echtes „Globales Gehirn" für die Menschheit verwandeln, das in der Lage wäre, jedes auch noch so vielschichtige Problem effizient zu lösen (Heylighen 1999, 2004).

Literatur

Berners-Lee, T. und Fischetti, M. (1999), *Weaving the Web. The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*, HarperCollins, San Francisco.

Bonabeau, E., Dorigo, M. und Theraulaz, G. (1999), *Swarm Intelligence: From Natural to Artificial Systems*, Oxford University Press, Oxford.

Buss, D. M. (1995), 'Evolutionary Psychology: A New Paradigm for Psychological Science', *Psychological Inquiry* **6**(1), S. 1–30. http://homepage.psy.utexas.edu/homepage/group/BussLAB/pdffiles/ANewParadigmforPsych.PDF [12. Feb 2007].

De Long, J. B. und Froomkin, A. M. (1998), The Next Economy?, *in* D. Hurley, B. Kahin und H. Varian (Hrsg.), 'Internet Publishing and Beyond: The Economics of Digital Information and Intellectual Property', MIT Press, Cambridge.

Elliot, M. (2006), 'Stigmergic Collaboration: The Evolution of Group Work', *M/C Journal* **9**(2). http://journal.media-culture.org.au/0605/03-elliott.php [22. Jan 2007].

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.dk/issues/issue3_3/ghosh/index.html [22. Jan 2007].

Grassé, P.-P. (1959), 'La reconstruction du nid et les coordinations inter-individuelles chez bellicositermes natalensis et cubitermes sp. la theorie de la stigmergie', *Insectes Sociaux* **6**, S. 41–81.

Hajjem, C., Harnad, S. und Gingras, Y. (2005), 'Ten-Year Cross-Disciplinary Comparison of the Growth of Open Access and How it Increases Research Citation Impact', *IEEE Data Engineering Bulletin* **28**(4), S. 39–47. http://sites.computer.org/debull/A05dec/hajjem.pdf.

Heylighen, F. (1994), World-Wide Web: A Distributed Hypermedia Paradigm for Global Networking, *in* 'Proceedings of SHARE Europe, Spring 1994, The Changing Role of IT in Business', Genf, S. 355–368.

Heylighen, F. (1999), 'Collective Intelligence and its Implementation on the Web', *Computational and Mathematical Theory of Organizations* **5**(3), S. 253–280.

Heylighen, F. (2004), Das Globale Gehirn als neue Utopia, *in* R. Maresch und F. Rötzer (Hrsg.), 'Renaissance der Utopie', Suhrkamp, Frankfurt am Main.

Heylighen, F. (2007), Mediator Evolution, *in* D. Aerts, B. D'Hooghe und N. Note (Hrsg.), 'Worldviews, Science and Us', World Scientific, Singapore.

Heylighen, F. und Gershenson, C. (2003), 'The Meaning of Self-organization in Computing', *IEEE Intelligent Systems* **18**(4), S. 72–75.

Howcroft, D. (2001), 'After the Goldrush: Deconstructing the Myths of the Dot.com Market', *Journal of Information Technology* **16**(4), S. 195–204.

Lerner, J. und Tirole, J. (2002), 'Some Simple Economics of Open Source', *Journal of Industrial Economics* **50**(2), S. 197–234.

Lih, A. (2004), Wikipedia as Participatory Journalism: Reliable Sources?, *in* '5th International Symposium on Online Journalism'. http://jmsc.hku.hk/faculty/alih/publications/utaustin-2004-wikipedia-rc2.pdf [22. Jan 2007].

Martens, B. (2004), *The Cognitive Mechanics of Economic Development and Institutional Change*, Routledge, Oxford.

Maslow, A. H. (1970), *Motivation and Personality*, 2. Aufl., Harper & Row, New York.

Moody, G. (2002), *Rebel Code: Linux and the Open Source Revolution*, Perseus Publishing, Cambridge, MA, USA.

Muffatto, M. und Faldani, M. (2003), 'Open Source as a Complex Adaptive System', *Emergence* **5**(3), S. 83–100.

Raymond, E. S. (1999), *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*, O'Reilly & Associates, Cambridge, MA, USA.

Reber, A. S. (1993), *Implicit Learning and Tacit Knowledge*, Oxford University Press, New York.

Robles, G., Merelo, J. J. und Gonzalez-Barahona, J. M. (2005), Self-Organized Development in Libre Software: A Model based on the Stigmergy Concept, *in* '6th International Workshop on Software Process Simulation and Modeling', St. Louis, MI, USA.

Simon, H. A. (1971), Designing Organizations for an Information-Rich World, *in* M. Greenberger (Hrsg.), 'Computers, Communication, and the Public Interest', Johns Hopkins Press, Baltimore, MD, USA.

Susi, T. und Ziemke, T. (2001), 'Social Cognition, Artefacts, and Stigmergy', *Cognitive Systems Research* **2**(4), S. 273–290.

Weber, S. (2004), *The Success of Open Source*, Harvard University Press, Cambridge, MA und London.

Williamson, O. E. und Masten, S. E. (1995), *Transaction cost economics*, Edward Elgar.

van Wendel de Joode, R. (2004), 'Innovation in Open Source Communities Through Processes of Variation and Selection', *Knowledge, Technology & Policy* **16**(4), S. 30.

von Hayek, F. A. (1945), 'The Use of Knowledge in Society', *American Economic Review* **35**(4), S. 519–530.

Kapitel 3

Open Source: Die Entscheidung

„Es gibt keine falschen Entscheidungen, es gibt nur Konsequenzen."

– *Rainer Sauer*

Die Entscheidung: Einleitung

HANS HENNING

(CC-Lizenz siehe Seite 563)

Open-Source-Software (OSS) hat sich im Laufe der letzten Jahre zu einer ernst zu nehmenden Alternative zu proprietären Softwarelösungen entwickelt. Die Entscheidung über den Einsatz von Open Source sollte aber dennoch wohl überlegt sein. Das folgende Kapitel widmet sich verschiedenen Aspekten, die bei dieser Entscheidung bedacht werden sollten und bietet darüber hinaus in Form von Erfahrungsberichten Einblicke in die Geschichten erfolgreicher Open-Source-Migrationen.

Den Anfang machen Jürgen Bitzer und Philip J. H. Schröder, die in „Entwicklung von Open-Source-Software: Kostenrelevante Eigenschaften einer ungewöhnlichen Organisationsform" die Kostenseite von OSS-Entwicklungen untersuchen. Dabei vergleichen sie die Entwicklung von OSS mit der von proprietärer Software anhand von Kosten- und Effizienzkriterien und räumen dabei einige verbreitete Vorurteile über das OSS-Entwicklungsmodell aus.

Einen ähnlichen Weg schlägt der Artikel „Open-Source-Software – Einsatzstrategien, Reifegrad und Wirtschaftlichkeit" von Michael Vetter, Thomas Renner, Holger Kett und Sascha Rex ein. Hier werden basierend auf einer Studie der Fraunhofer–Gesellschaft die Einsatzstrategien und -potenziale von OSS analysiert und mit Hilfe eines Beispiels die Wirtschaftlichkeit von OSS anhand der *Total Cost of Ownership* (TCO) untersucht.

Die Brücke von der Theorie zur Praxis schlägt der Artikel von Hal Varian und Carl Shapiro, die in „Einführung von Linux im öffentlichen Sektor" die Vor- und Nachteile des Einsatzes von OSS und proprietärer Software in öffentlichen Verwaltungen, aber auch in anderen Umgebungen untersuchen und gegenüberstellen. Dabei werden auch die unterschiedlichen Aspekte offener und geschlossener Standards und die daraus entstehenden Gefahren von Lock-in-Effekten sowie die positiven Auswirkungen, die der breite Einsatz von OSS in der Verwaltung auf die Volkswirtschaft haben kann, behandelt.

Der darauf folgende Artikel widmet sich dann der Praxis und der Politik: Stefan Krempl berichtet in „Linux fensterlt in der Verwaltung" über verschiedene Migrationsprojekte zur Einführung von Open Source im öffentlichen Sektor und geht an den

Beispielen der Münchner Stadtverwaltung[1] und des Auswärtigen Amtes auch kritisch auf das Thema Lobbyismus ein.

Die Migration des Auswärtigen Amtes wird von Torsten Werner genauer beleuchtet: Unter dem Titel „World Domination" berichtet er über „Die Erfolgsgeschichte der Linux- und Open-Source-Einführung im Auswärtigen Amt" und behandelt dabei auch die speziellen Anforderungen, die durch die weite geographische Verteilung und die Menge der Standorte bei dieser Migration entstanden sind.

Den Abschluss bildet ein dritter Praxisartikel: Carsten Brunke und Jörg Meyer berichten in „Nach der Migration – business as usual?" von ihren Erfahrungen mit den IT-Systemen der Norddeutschen Affinerie AG nach der Servermigration auf *Samba* und *OpenLDAP* und greifen damit ihren im Open Source Jahrbuch 2005 erschienenen Artikel „Migration auf Samba/OpenLDAP bei der Norddeutschen Affinerie AG" auf. Neben der grundlegenden Frage, wie sich die neuen Systeme in der Praxis bewährt haben, behandeln die Autoren auch Kosten- und Verfügbarkeitsaspekte der Systeme nach der Migration sowie weitere Veränderungen, die seither an diesen vorgenommen wurden.

Alle Artikel dieses Kapitels unterstreichen, dass Open Source gegenüber proprietärer Software in den meisten Fällen zumindest gleichwertig ist. Es gibt im IT-Bereich nur wenige Patentlösungen und die Entscheidung für oder gegen Open Source muss daher von Fall zu Fall individuell geklärt werden. Fest steht jedoch eines: Ob es nun um Effizienz- oder Kostenfragen geht oder die Frage nach bestimmten Einsatzpotenzialen im Vordergrund steht, OSS ist nicht mehr nur eine Spielerei von und für Geeks, sondern eine ernst zu nehmende Alternative, gerade auch für den kommerziellen Einsatz. Auch das OSS-Entwicklungsmodell hat inzwischen einen festen Platz in der Softwareentwicklung. Firmen wie *RedHat* und *Novell* zeigen, dass sich mit den unorthodoxen Entwicklungsmethoden verteilter Open-Source-Entwickler auch Geld machen lässt. Kurz ausgedrückt: Open Source funktioniert.

1 Siehe hierzu auch den Artikel „Das Projekt LiMux – Freie Software für die Münchner Verwaltungsclients" von Wilhelm Hoegner im Open Source Jahrbuch 2006.

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

JÜRGEN BITZER UND PHILIPP J. H. SCHRÖDER

(CC-Lizenz siehe Seite 563)

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

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

1 Einleitung

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

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

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

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

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

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

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

2 OSS-Entwicklung: eine ökonomische Perspektive

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

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

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

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

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

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

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

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

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

3 Kostensparende Eigenschaften des OSS-Entwicklungsmodells

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

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

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

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

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

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

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

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

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

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

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

4 Kostenverursachende Eigenschaften des OSS-Entwicklungsprozesses

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

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

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

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

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

5 Zusammenfassung und Ausblick

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

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

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

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

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

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

Literatur

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Open-Source-Software – Einsatzstrategien, Reifegrad und Wirtschaftlichkeit

MICHAEL VETTER, THOMAS RENNER, HOLGER KETT
UND SASCHA REX

(CC-Lizenz siehe Seite 563)

Open-Source-Software stellt für Unternehmen und die öffentliche Hand eine Alternative zu kommerziellen Softwareprodukten dar. Die Fraunhofer–Gesellschaft untersuchte 2005 im Rahmen einer Studie Einsatzpotenziale und Wirtschaftlichkeit von Open-Source-Software. Dieser Beitrag stellt ausgewählte Ergebnisse vor. Es wird anhand einzelner Anwendungsfelder aufgezeigt, wie Reifegrad und Relevanz der Open-Source-Software eingeschätzt werden und welche Einsatzstrategien sich daraus ergeben. Anhand eines Beispielszenarios, der Migration von Arbeitsplatzrechnern auf Open-Source–Software, werden die Ergebnisse einer Wirtschaftlichkeitsanalyse auf Basis der *Total Cost of Ownership* dargestellt.

Schlüsselwörter: Reifegrad · TCO · Portfolioanalyse · Migration

1 Einleitung

Diverse Studien und Migrationsprojekte in Unternehmen und öffentlichen Institutionen zeigen, dass Open-Source-Software mittlerweile nicht mehr nur auf dem Server, sondern auch auf dem Desktop eine ernsthafte Konkurrenz für kommerzielle Softwareprodukte geworden ist.

Das *Competence Center Electronic Business* am Fraunhofer-Institut für Arbeitswirtschaft und Organisation führte in den Jahren 2004 und 2005 für die Fraunhofer-Gesellschaft eine interne Studie zum Einsatz von Open-Source-Software durch, die in wesentlichen Teilen auch extern veröffentlicht wurde.[1] Der vorliegende Beitrag

1 Die Studie (Renner et al. 2005) ist kostenlos verfügbar unter http://www.e-business.iao.fraunhofer.de/docs/fhg_oss-studie.pdf.

fasst Ergebnisse aus dieser Studie zusammen. Die Anforderungen und Rahmenbedingungen der Fraunhofer-Gesellschaft sind zwar nicht identisch mit denen von Unternehmen, aber die Methodik und Teile der Ergebnisse sind übertragbar.

Die Studie stellt Einsatzpotenziale von Open-Source-Software in den wichtigsten Teilbereichen dar und zeigt eine Methodik, um die Wirtschaftlichkeit des Einsatzes zu berechnen. Der derzeitige Stand des Einsatzes freier Software wird beispielhaft anhand einer Erhebung dargestellt, die in der Fraunhofer-Gesellschaft durchgeführt wurde.

Des Weiteren werden für wichtige Softwarearten Vergleichsbetrachtungen (Stand Juni 2005) durchgeführt, in deren Rahmen freie und kommerzielle Software einander gegenübergestellt werden. Die Studie hatte nicht das Ziel, eine vollständige Markterhebung für jegliche Art von Software darzustellen. Vielmehr werden generelle Tendenzen herausgearbeitet, die als Leitlinie für eigene Entscheidungen dienen können.

Die Fraunhofer-Gesellschaft ist eine öffentliche Großforschungseinrichtung, die aus mehr als 50 einzelnen Einrichtungen mit zusammen über 12 000 Mitarbeitern besteht. Wichtigstes Tätigkeitsgebiet der Fraunhofer-Gesellschaft ist die anwendungsorientierte Forschung, sowohl für öffentliche als auch private Auftraggeber. Die Zielsetzungen dieser Studie sind im Folgenden aufgeführt:

Untersuchung verbreiteter Open-Source-basierter Lösungen Im November 2004 wurde eine Erhebung bei IT-Verantwortlichen und Projektleitern der Fraunhofer-Gesellschaft durchgeführt, die insbesondere folgende Fragestellungen beleuchtete:

- Klärung der Relevanz verschiedener Softwarearten in einer Großforschungseinrichtung
- Stellung und Verbreitungsgrad von Open-Source-Software in den jeweiligen Bereichen
- Reife von Open-Source-Software
- Betrachtung von Vor- und Nachteilen freier Software
- Erhebung des Partizipationsgrades am Entwicklungsprozess von freier Software

Als Grundlage für die Befragung wurde ein standardisierter Fragebogen entwickelt, der im Wesentlichen aus geschlossenen Fragen aufgebaut war. Es wurden insgesamt 55 Fragebögen ausgegeben. Ausgewertet werden konnten 42 Fragebögen, was einer Rücklaufquote von 76 Prozent entspricht. Die Ergebnisse dieser Erhebung sind in der Studie detailliert dargestellt.

Darstellung von Open-Source-Lösungen Für die relevanten Anwendungsgebiete wurden wichtige kommerzielle und freie Softwareprodukte gegenübergestellt, und dabei diskutiert, welche Einsatzgebiete und -szenarien sich für den Einsatz von Open-Source-Produkten eher eignen und welche weniger.

Wirtschaftlichkeitsbetrachtung Für einige ausgewählte Einsatzgebiete wurden exemplarisch Wege zur Durchführung von Wirtschaftlichkeitsbetrachtungen bzw. Kosten/Nutzen-Analysen aufgezeigt. Die Ergebnisse lassen sich für konkrete Szenarien leicht erweitern bzw. vervollständigen.

Ausarbeitung von Empfehlungen Es wurden strategische Empfehlungen ausgearbeitet, sowohl zum Einsatz von Open-Source-Software als auch zu weitergehenden Fragestellungen.

2 Reife von Open-Source-Software

In der Studie wurde die Einstellung der Befragten zur Reife von Open-Source–Software abgefragt. Die Reife von Software wurde anhand der folgenden aus dem FURPS-Modell abgeleiteten fünf Indikatoren für Softwarequalität gemessen.

- *Functionality* (Funktionalität)
- *Usability* (Benutzerfreundlichkeit)
- *Reliability* (Zuverlässigkeit)
- *Performance* (Leistung)
- *Support*

Den Befragten wurde dazu eine fünfstufige Ratingskala zur Bewertung des jeweiligen Reifegrades vorgelegt (1 = Reifegrad sehr niedrig; 5 = Reifegrad sehr hoch). Der Reifegrad sollte relativ zu relevanten kommerziellen Softwareprodukten betrachtet werden. Ziel der Frage war es, eine Aussage über die Ausgereiftheit von Open–Source-Software untergliedert nach Softwarearten machen zu können. Dies ist ein erster Anhaltspunkt, in welchen Bereichen über den Einsatz von Open-Source-Software nachgedacht werden kann und sollte.

2.1 Reifegrad der Funktionalität

In Bezug auf die Funktionalität wurden die Open-Source-Betriebssysteme als am ausgereiftesten betrachtet (Messwert 4,4). Auch den freien Entwicklungssystemen und Datenbanken wird eine sehr hohe funktionale Reife zugesprochen (4,1 bzw. 4,0). Das recht breite Mittelfeld wird von den Office-Paketen angeführt (3,8) und reicht bis zur Groupware (3,2). Noch akzeptable Werte erreichte Open-Source-Software in den Bereichen Application-Service-Providing-Software und Dokumenten-Management–Systeme (jeweils 2,9). Als funktional nicht sehr ausgereift wurden freie CRM-/SRM–Systeme, Videosoftware, ERP-Systeme und Sprachverarbeitung betrachtet (Messwerte zwischen 2,5 und 2,3).

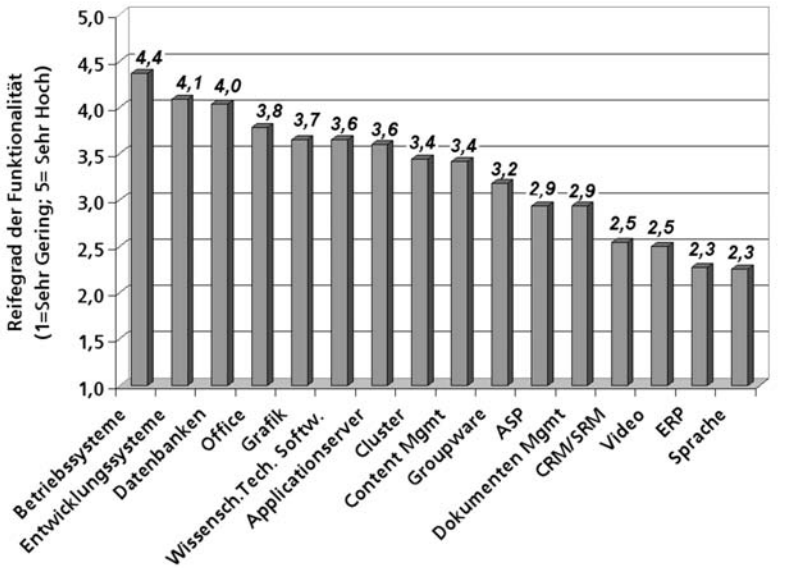


Abbildung 1: Reifegrad der Funktionalität

2.2 Reifegrad der Benutzerfreundlichkeit

Am benutzerfreundlichsten ist Open-Source-Software nach der Ansicht der Befragten in den Bereichen Office-Produkte sowie Betriebssysteme. Beide Softwarearten erreichten einen Messwert von 3,9. Auch freie Entwicklungssysteme und Datenbanken können mit kommerzieller Software gut mithalten. Diese erreichten einen Messwert von jeweils 3,7. Der Spitzengruppe sind ferner noch die Grafiksoftware sowie die wissenschaftlich-technische Software zuzurechnen (3,5 bzw. 3,4). Mittlere Benutzerfreundlichkeit haben nach Ansicht der Befragten die freien Application-Server, Groupware-Applikationen, Content-Management-Systeme sowie die Cluster- und ASP-Lösungen zu bieten (Messwerte von 3,1 bis 2,9). Am wenigsten anwenderfreundlich wurden Open-Source-CRM-/SRM-Lösungen, Dokumenten-Management-Systeme sowie Video- und Sprachapplikationen gesehen (2,7 bis 2,4). Schlusslicht sind ERP-Systeme mit einem Wert von 2,3.

2.3 Reifegrad der Zuverlässigkeit

Als relativ zu kommerziellen Produkten am zuverlässigsten und sichersten wurden die freien Betriebssysteme, Datenbanken und Entwicklungssysteme betrachtet (Messwerte 4,5 bis 4,3). Ebenfalls noch gute Werte erreichten in dieser Kategorie die

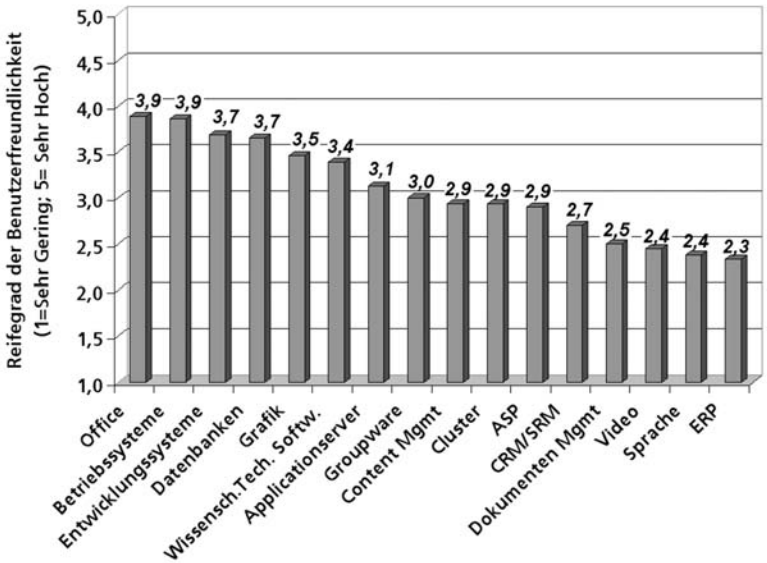


Abbildung 2: Reifegrad der Benutzerfreundlichkeit

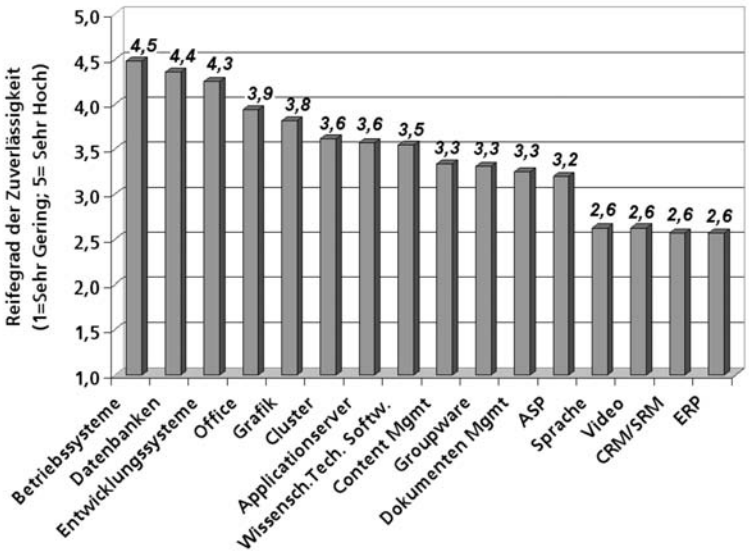


Abbildung 3: Reifegrad der Zuverlässigkeit

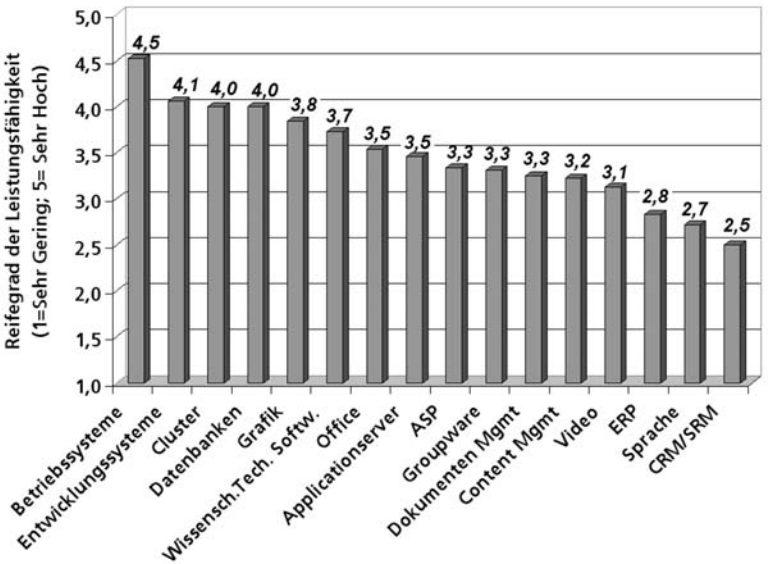


Abbildung 4: Reifegrad der Leistungsfähigkeit

Office-Produkte und Grafiksoftware (3,9 und 3,8). Mittlere Werte konnten Open–Source-Produkte von dem Bereich Clustersoftware bis hin zu ASP-Lösungen erreichen (Messwerte zwischen 3,6 und 3,2). Als relativ wenig zuverlässig und sicher wurde freie Software der Bereiche Sprachapplikationen, Videosoftware, CRM-/SRM- und ERP-Systeme gesehen (alle 2,6).

2.4 Reifegrad der Leistungsfähigkeit

In diesem Bereich gehen die freien Betriebssysteme als klarer Sieger im Open-Source–Markt hervor. Gemessen an kommerzieller Software erreichte die Leistungsfähigkeit der freien Alternativen einen Messwert von 4,5. Des Weiteren wurde auch den freien Entwicklungssystemen, Clusterlösungen und Datenbanken eine überdurchschnittlich hohe Leistungsfähigkeit bescheinigt (4,1 bis 4,0). Gute Werte erreichten ferner Grafiksoftware und wissenschaftlich-technische Software (3,8 bzw. 3,7). Das Mittelfeld beginnt hier bei Office-Produkten und endet bei Videosoftware – deren Leistungsfähigkeit erreichte Messwerte zwischen 3,5 und 3,1. Unterdurchschnittliche Leistungsfähigkeit wurden freien ERP-Systemen, Sprachapplikationen und CRM-/SRM-Systemen attestiert.

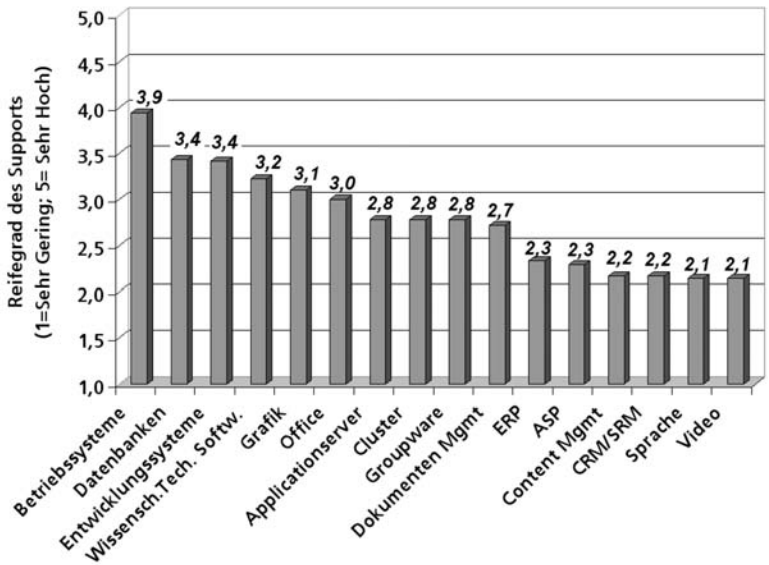


Abbildung 5: Reifegrad des Supports

2.5 Reifegrad des Supports

Auch in der Frage des Supports haben die freien Betriebssysteme die Nase vorn. Die relative Güte des Supports wurde hier mit 3,9 bewertet. Noch guter Support ist für freie Datenbanken und Entwicklungssysteme erhältlich (Messwert jeweils 3,4). Das Mittelfeld reicht von wissenschaftlich-technischer Software bis zu Dokumenten–Management-Systemen (Werte zwischen 3,2 und 2,7). Mit der kommerziellen Konkurrenz nicht mithalten kann der Support freier Software in den Bereichen ERP, ASP, Content-Management sowie CRM-/SRM-Systeme und Videosoftware (2,3 bis 2,1).

3 Portfolio-Analyse

In diesem Abschnitt werden generelle Empfehlungen bezüglich des Einsatzes von Open-Source-Software ausgesprochen. Dazu wurde aus den Daten der Ist-Analyse ein Relevanz-Reifegrad-Portfolio (Abbildung 6) abgeleitet, in dem sowohl die in der Befragung erhobene Bedeutung der einzelnen Einsatzgebiete[2] als auch die relative Reife der vorhandenen Open-Source-Software aufgezeichnet ist. Der Reifegrad lässt sich aus den folgenden fünf Teilkomponenten ermitteln:

2 Es wurden die Bedeutungswerte aus der in der Studie dargestellten Befragung zugrunde gelegt. Diese sind selbstverständlich nicht für alle Unternehmen identisch.

– Reifegrad der Funktionalität
– Reifegrad der Benutzerfreundlichkeit
– Reifegrad der Zuverlässigkeit / Sicherheit
– Reifegrad der Leistungsfähigkeit
– Reifegrad des Supports

Die Gewichtung wurde aus den Top-Two-Werten der Antworten auf die Frage nach der Wichtigkeit der fünf Teilkriterien abgeleitet. Es ergaben sich die in Tabelle 1 dargestellten Gewichtungen. Die Wichtigkeit der jeweiligen Softwarearten wurde in der Befragung direkt erhoben und konnte somit direkt in das Portfolio einfließen, das in vier Quadranten unterteilt wird, für die unterschiedliche Strategieempfehlungen gelten.

Kriterium	Gewicht
Zuverlässigkeit	0,39
Funktionalität	0,32
Benutzerfreundlichkeit	0,18
Leistung	0,07
Support	0,05

Tabelle 1: Gewichtungstabelle für den Reifegrad

3.1 Strategiegebiet I: Freie Software einsetzen und vorantreiben

Im oberen, rechten Quadranten des Portfolios eingetragene Einsatzgebiete weisen eine hohe Relevanz auf und sind gleichzeitig durch einen hohen Reifegrad von Open–Source-Software in diesem Bereich gekennzeichnet. In dieses Handlungsfeld fallen folgende Softwarearten:

– Betriebssysteme
– Office-Pakete
– Datenbanken
– Entwicklungswerkzeuge
– Wissenschaftlich-technische Software
– Grafiksoftware
– Application-Server
– Groupware

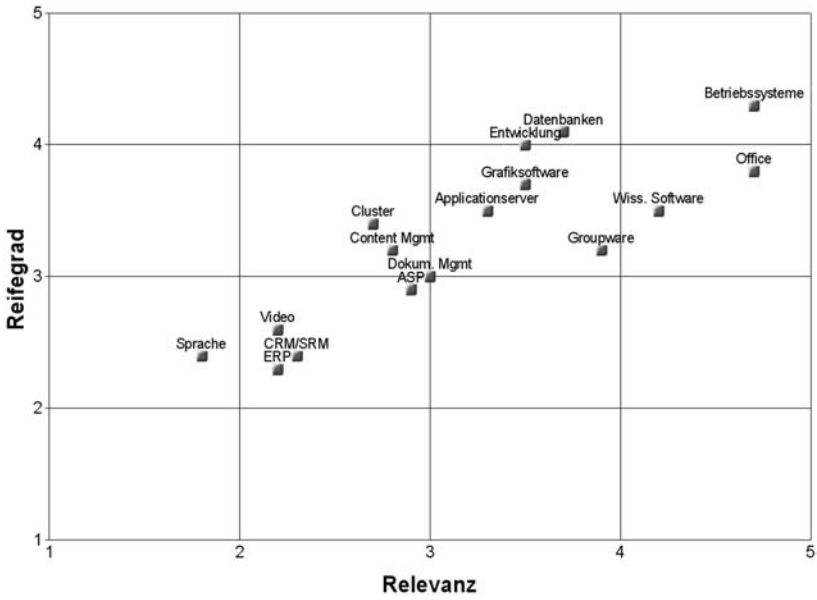


Abbildung 6: Das Relevanz-/Reifegrad-Portfolio von Open-Source-Software

Diese Softwarearten haben aufgrund ihrer Relevanz eine strategische Bedeutung. Viele dieser Softwarearten sind zudem durch hohe Einsatzhäufigkeiten gekennzeichnet. Das hohe Volumen wirkt sich wiederum auf das Einsparpotenzial durch den Einsatz freier Software aus. Für diese Gebiete sollten daher zeitnah weitere Untersuchungen durchgeführt werden. Diese sollten vor allem folgende Aspekte beleuchten:

- Detaillierte Aufnahme von Anforderungen an Software im jeweiligen Bereich
- Abgleich der Anforderungen mit vorhandener freier Software
- Konzeption von konkreten Einführungs- oder Migrationsszenarien
- Durchführung von detaillierten Wirtschaftlichkeits- und Nutzwertanalysen für jeden dieser Bereiche
- Erarbeitung von Leitfäden und Einsatzszenarien für freie Software
- Unter Umständen auch Zusammenstellung von Know-how und Erfahrungsberichten – dies könnte etwa auch den dauerhaften Betrieb einer unternehmensweiten Informationsaustauschplattform beinhalten.

Des Weiteren sind flankierende Maßnahmen zu überlegen, wie z. B. die Förderung konkreter Open-Source-Projekte in diesen Gebieten. Dies könnte sowohl eine

finanzielle oder ideelle Förderung als auch eine direkte Mitarbeit an den Entwicklungsarbeiten beinhalten. Somit würde es den Anwendern ermöglicht, eventuell noch benötigte Funktionen in die Software einzubringen. Gegebenenfalls kann auch eine Mitarbeit in übergeordneten Gremien und Verbänden erfolgen, die sich mit freier Software beschäftigen. Durch diese Mitarbeit können weitere Punkte identifiziert werden, in denen Handlungsbedarf besteht.

3.2 Strategiegebiet II: Freie Software einsetzen

Der obere, linke Quadrant kennzeichnet Einsatzgebiete mit hoher Verfügbarkeit von ausgereifter freier Software, aber gleichzeitig unterdurchschnittlicher Relevanz. In dieses Handlungsfeld fallen folgende Softwarearten:

- Web-Content-Management-Systeme
- Cluster- und Grid-Software

In diesen Fällen ist der Einsatz freier Software zu empfehlen. Diese Softwarearten sind jedoch tendenziell weniger bedeutsam, so dass auf weitergehende Maßnahmen eher verzichtet werden sollte. Zu den Möglichkeiten des Einsatzes sollten wie im Strategiegebiet I weitergehende Untersuchungen durchgeführt werden.

3.3 Strategiegebiet III: Freie Software vorantreiben

Der untere, rechte Quadrant identifiziert Einsatzgebiete mit hoher Relevanz und gleichzeitigem Fehlen ausgereifter Open-Source-Software. Hier handelt es sich um Einsatzgebiete mit hoher strategischer Relevanz und hohen Einsparpotenzialen. Die Portfolioanalyse zeigte keine Softwareart auf, die deutlich in diese Kategorie fällt. Am ehesten ist der Bereich Dokumenten-Management-Systeme hierunter zu sehen. Aufgrund der recht hohen strategischen Bedeutung dieser Software wäre zu überlegen, geeignete Open-Source-Projekte zu identifizieren und deren Software durch gezielte Maßnahmen zur Einsatzreife zu verhelfen.

3.4 Strategiegebiet IV: Kein Handlungsbedarf

Der untere, linke Quadrant kennzeichnet Softwarearten mit geringerer Bedeutung und gleichzeitigem Fehlen von ausgereifter Open-Source-Software. In dieses Handlungsfeld fallen folgende Softwarearten:

- Application-Service-Providing-Software
- Videosoftware
- Enterprise-Ressource-Planning-Systeme
- CRM-/SRM-Systeme

– Sprachverarbeitung

Die strategische Bedeutung dieser Einsatzgebiete ist für die untersuchte Institution gering, die Einsparpotenziale sind vermutlich begrenzt. Hierbei ist zu beachten, dass sich die niedrige Relevanz von Enterprise-Ressource-Planning-Systemen sowie Customer- und Supplier-Relationship-Management (CRM/SRM) auf den Einsatz in Kundenprojekten der Fraunhofer-Institute bezieht. Diese Bewertung kann nicht auf alle Unternehmen übertragen werden.

Gleichzeitig ist vorhandene Open-Source-Software im Regelfall nicht ausgereift genug, um einen Einsatz oder eine Migration empfehlen zu können. Für diese Einsatzgebiete ist es daher zu empfehlen, vorläufig weiterhin kommerzielle Software einzusetzen. Gleichwohl sollten auch diese Bereiche beobachtet werden, da es in fast allen genannten Gebieten vielversprechende Open-Source-Projekte gibt, deren Software eines Tages auch die Anforderungen abdecken könnte.

4 Wirtschaftlichkeit

Die Untersuchung der Wirtschaftlichkeit des Einsatzes von Open-Source-Software bildet – neben strategischen Überlegungen und den konkreten funktionalen und technischen Anforderungen, die erfüllt werden müssen – ein sehr wichtiges Auswahl- und Entscheidungskriterium. Eine ausführliche Methodik zur Wirtschaftlichkeitsberechnung und Ermittlung der *Total Cost of Ownership* (TCO) ist in der vollständigen Studie dargestellt.

Aufgrund der Vielfalt und der unterschiedlichen Bedeutung der Open-Source–Anwendungsfelder für die Fraunhofer-Gesellschaft wurden die drei wichtigsten Anwendungsbereiche für eine nähere Wirtschaftlichkeitsbetrachtung ausgewählt. Diese Bereiche ergaben sich aus der Befragung der IT-Leiter der Fraunhofer-Gesellschaft nach der Relevanz von Einsatzgebieten. Als Ergebnis wurden die Anwendungsfelder Betriebssysteme, Office- und Groupware-Anwendungen als besonders relevant für die Fraunhofer-Gesellschaft identifiziert. Im Folgenden wird eine grobe Abschätzung des Potenzials bei einer vollständigen Migration von Betriebssystem, Office-Anwendung und Groupware für die persönlichen Arbeitsplatzrechner von Fraunhofer-Mitarbeitern dargestellt.

4.1 Szenario: Migration persönlicher Arbeitsplatzrechner von Mitarbeitern

Die Wirtschaftlichkeitsbetrachtung erfolgt anhand eines konkreten Szenarios. Der Kern des Szenarios ist die Migration von persönlichen Mitarbeiterrechnern mit aktueller Software von *Microsoft* (Betriebssystem, Office-Software, *Outlook*) auf Open–Source-Software. Hierfür werden auf Basis einer Total-Cost-of-Ownership-Analyse (TCO-Analyse) alle relevanten Kostentreiber ermittelt und monetär bewertet. Eine Prüfung, ob Open-Source-Software in den besagten Anwendungsfeldern für eine

Migration geeignet ist, wurde zuvor durchgeführt und wird im weiteren Verlauf der Wirtschaftlichkeitsbetrachtung als gegeben vorausgesetzt.

Grundlage der Bewertung waren konkrete Informationen aus den Bereichen Kostenrechnung, Beschaffung, Personalwirtschaft und mehrerer IT-Abteilungen einzelner Fraunhofer-Institute. Die Informationen stammten überwiegend aus telefonischen und persönlichen Befragungen der verantwortlichen Mitarbeiter.

4.2 Ergebnis der Wirtschaftlichkeitsbetrachtung

Die Kosten für die Fortführung einer Microsoft-basierten und einer reinen Open–Source-basierten IT-Infrastruktur wurden anhand der Kapitalwertmethode aufbereitet und zum Vergleich gegenübergestellt. Hierbei werden zum einen die Vollkosten und zum anderen nur die haushaltswirksamen Teilkosten verglichen. Bei einem Vollkostenvergleich ergibt sich ein Einsparpotenzial von ca. 2,4 Prozent. Betrachtet man jedoch nur die wirklich haushaltswirksamen Kosten (Kosten, die erst aufgrund des zu betrachtenden Vorhabens entstehen), so lassen sich ca. 6,9 Prozent der zusätzlich zu den schon vorhandenen Kosten aufzubringenden Mittel einsparen. Dies stellt eine erste Abschätzung des Einsparpotenzials dar.

Der Wegfall der Lizenzkosten durch den Einsatz von Open-Source-Software führt in vielen Fällen zu einer Senkung der TCO von Software. Die kommerziellen Anbieter bestreiten diesen Umstand teilweise heftig und argumentieren mit dem relativ geringen Anteil der Lizenzkosten an den TCO und den hohen Schulungskosten, die ihrer Meinung nach bei einer Migration hin zu Open-Source-Software anfallen.

Trotzdem konnte von vielen Studien eine geringere TCO von Open-Source-Software bestätigt werden. So stellte *Cybersource* in einer Vergleichsstudie zwischen den TCO von *Linux* und *Windows* eine zwischen 19 Prozent und 36 Prozent niedrigere TCO von *Linux* fest (Cybersource 2002).

Die Wirtschaftlichkeit von Open-Source-Software hängt stark vom jeweiligen Einsatzgebiet ab. Die TCO von Open-Source-Software ist im Vergleich zu proprietärer Software insbesondere dann deutlich niedriger, wenn wenig zusätzlicher Schulungsaufwand anfällt und gleichzeitig große Lizenzkostenblöcke eingespart werden können. *Soreon* stellte daher insbesondere bei Großunternehmen mit hohen Lizenzkosten Einsparpotenziale bis zu 30 Prozent fest (Binder und Henke 2003). Finanziell besonders attraktiv sind freie Office-Pakete,Serverbetriebssysteme, Content-Management-Systeme und Datenbanken.

Der Vergleich zwischen einer Linux- und einer Windows-basierten Infrastruktur im Zuge der Migrationsüberlegungen der Stadt München ergab, dass für die rund 14 000 Arbeitsplätze die Kosten beider Lösungen annähernd identisch waren – *Microsoft* machte dabei erhebliche Zugeständnisse an die prestigeträchtige bayerische Landeshauptstadt (Unilog Integrata 2004).

Dennoch ist festzuhalten, dass die Einsparpotenziale stark mit dem jeweiligen Einsatzszenario zusammenhängen. Während Open-Source-Software nur in seltenen

Fällen insgesamt deutlich teurer als kommerzielle Software sein wird, so lässt sich doch nicht in jedem Fall eine drastische Senkung der TCO realisieren. Wichtig ist deshalb, eine konkrete Wirtschaftlichkeitsbetrachtung für die geplanten Einsatzszenarien vorzunehmen. Dass dies auch für kommerzielle Software seltener als nötig durchgeführt wird, darf an dieser Stelle kein Argument sein.

Das in der Fraunhofer-Studie dargestellte Vorgehen zur TCO-Analyse kann als Leitlinie für Wirtschaftlichkeitsbetrachtungen für Software dienen. Die Methodik lässt sich auch auf andere Migrations- oder Einführungsszenarien übertragen.

5 Zusammenfassung

Open-Source-Software ist in einigen Einsatzbereichen eine Alternative zu herkömmlicher kommerzieller Software. Freie Betriebssysteme, Office-Pakete, Entwicklungssysteme, Datenbanken und Content-Management-Systeme – um nur einige Beispiele zu nennen – erfüllen die qualitativen Anforderungen meist ebenso gut wie die kommerziellen Produkte.

Kostenaspekte sind ein Kriterium, das tendenziell eher für den Einsatz freier Software spricht, aber nicht das einzige. Darüber hinaus bringt der Einsatz freier Software auch strategische Vorteile mit sich. Diese sollten gegenüber dem – zweifellos wichtigen – Kriterium der Wirtschaftlichkeit nicht vernachlässigt werden.

Wichtig ist eine genaue Untersuchung für das jeweilige Einsatzgebiet, da Open–Source-Software stark unterschiedliche Reifegrade aufweist. Derzeit gibt es einige Softwarebereiche, in denen noch keine ausreichend ausgereiften Open-Source-Produkte existieren. Die Entwicklung lässt aber den Schluss zu, dass die Anzahl dieser Fälle zukünftig abnehmen wird. In einigen Bereichen, in denen Open-Source-Software derzeit noch hinter den kommerziellen Produkten zurückfällt, sind viel versprechende Open-Source-Projekte aktiv. In strategisch wichtigen Einsatzgebieten ist die Förderung entsprechender Open-Source-Projekte zu empfehlen.

Literatur

Binder, S. und Henke, M. (2003), *Kassensturz. Open Source und proprietäre Software im Vergleich*, Soreon Research, Frankfurt.

Cybersource (2002), Linux vs. Windows: Total Cost of Ownership Comparison, Studie, Cybersource. http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf [21. Jan 2007].

Renner, T., Vetter, M., Rex, S. und Kett, H. (2005), *Open Source Software – Einsatzpotenziale und Wirtschaftlichkeit*, Fraunhofer IRB Verlag, Stuttgart. http://www.e-business.iao.fraunhofer.de/docs/fhg_oss-studie.pdf [21. Jan 2007].

Unilog Integrata (2004), Client-Studie der Landeshauptstadt München, Studie, Kurzfassung, München. http://www.muenchen.info/pia/clientstudie_kurz.pdf [21. Jan 2007].

Einführung von Linux im öffentlichen Sektor*

HAL VARIAN UND CARL SHAPIRO

(CC-Lizenz siehe Seite 563)

Dieser Artikel untersucht die Möglichkeiten eines Einsatzes von Linux, hauptsächlich für den öffentlichen Sektor. Dabei wird deutlich, dass das freie Betriebssystem mit gängiger proprietärer Software konkurrieren kann. Es verfügt über eine große Anzahl interessanter Eigenschaften, wie z. B. die Möglichkeit für die Anwender, Lock-In-Effekten zu entgehen oder die grundsätzliche Nutzung von offenen Schnittstellen. Darüber hinaus gibt es besonders für den öffentlichen Sektor Gründe für eine breite Nutzung von Open-Source-Software. Dies könnte positive Auswirkungen auf die gesamte Softwarebranche eines Landes haben und so insgesamt zu wirtschaftlichen Vorteilen führen.

Schlüsselwörter: Lock-In-Effekte · öffentliche Verwaltung · Softwarequalität · offene Schnittstellen

1 Einführung

Das Linux-Betriebssystem stellt in der Informationstechnologie, sowohl im privaten als auch im öffentlichen Sektor, zunehmend eine interessante Alternative dar, vor allem für den Betrieb von leistungsstarken Servern.[1] Die Entscheidung zur Umstellung auf eine andere Software-Plattform hat generell nachhaltige Auswirkungen auf die nachfolgende Einführung von Anwendungssoftware oder plattformbezogener Soft-

* Aus dem Englischen von Annett Wendschuh.

1 Server stellen Endnutzern Informationen oder Dienste bereit und arbeiten üblicherweise ohne ständige Überwachung. Server-Hardware können Großrechner (*Mainframes*), Arbeitsplatzsysteme (*Workstations*) oder Arbeitsplatzrechner (PCs) sein, wobei die Wahl von der Größenordnung und den Betriebsanforderungen des Einsatzgebietes abhängig ist. Im Gegensatz dazu versteht man unter Desktop–Computern die normalen Arbeitsplatzrechner (PCs), die für unterschiedliche interaktive Aufgaben wie Textverarbeitung, Kalkulation, E-Mail-Bearbeitung oder Internetdienste eingesetzt werden.

ware.[2] Noch umfassender und komplexer werden die Auswirkungen der Einführung von Linux durch die Tatsache, dass es sich um Open-Source-Software (OSS) statt um proprietäre Software handelt.

Darüber hinaus kann die umfassende Einführung von OSS-Plattformen im öffentlichen Sektor starken Einfluss auf die wirtschaftliche Entwicklung der gesamten Softwareindustrie eines Landes haben, ein Gesichtspunkt, der vor allem für öffentliche Entscheidungsträger von zentraler Bedeutung sein dürfte.

Obwohl sich diese Erörterung vorwiegend mit Linux beschäftigt, wird sie notwendigerweise auch auf die Wirtschaft der Softwaremärkte und auf die Unterschiede eingehen, die zwischen dem traditionellen Entwicklungsmodell für proprietäre Software und dem Modell für OSS bestehen, das von Linux, *Apache* und anderen bekannten Open-Source-Anwendungen genutzt wird. Hintergrundinformationen zu den wirtschaftlichen Gegebenheiten auf Softwaremärkten enthält der diesbezügliche Artikel im vorliegenden Band.[3] Für eine tiefgründigere Einarbeitung in diese Konzepte empfiehlt sich die Lektüre unseres Buches (Shapiro und Varian 1999).

Der vorliegende Artikel ist folgendermaßen aufgebaut: Zunächst wird ein kurzer Überblick über die grundsätzlichen Untersuchungsergebnisse und Empfehlungen gegeben. Der zweite Abschnitt beschäftigt sich mit der Definition und Erklärung der Grundkonzepte von OSS und proprietärer Software sowie offenen und proprietären Schnittstellen. Dabei wird besonderer Wert auf die Feststellung gelegt, dass OSS mit proprietärer Software interagiert und diese ergänzt. Außerdem sollen einige der grundlegenden wirtschaftlichen Bedingungen genauer bestimmt werden, die sich günstig auf das Open-Source-Modell für die Softwareentwicklung auswirken. Im dritten Teil werden die vorher beschriebenen Grundprinzipien angewendet, um Nutzen und Kosten der Einführung von Linux aus Sicht des Anwenders zu bewerten. Die Ergebnisse dieser Analyse gelten für private wie öffentliche Anwender gleichermaßen. Zum Abschluss beschäftigen wir uns kurz mit einigen zusätzlichen Überlegungen zur Rolle der öffentlichen Entscheidungsträger, insbesondere im Hinblick auf die Auswirkungen, die die Wahl einer bestimmten Software-Plattform auf die wirtschaftliche Entwicklung der Softwareindustrie eines Landes hat.

2 Untersuchungsergebnisse und Empfehlungen

Die Untersuchung kommt zu den folgenden grundsätzlichen Ergebnissen und Empfehlungen:

Kritische Masse Das Linux-Betriebssystem hat inzwischen eine kritische Masse erreicht. Dadurch kann den Anwendern die Sicherheit geboten werden, dass das Sys-

2 *Software-Plattformen*, zu denen Betriebssysteme zählen, dienen der Bereitstellung unterschiedlicher Dienste für Anwendungssoftware, welche ihrerseits geschrieben wird, um auf einer Software-Plattform ausgeführt zu werden.

3 Siehe S. 125 für den Beitrag „Die Ökonomie der Softwaremärkte".

tem auch zukünftig verfügbar bleiben und ständig verbessert werden wird, wodurch sich das finanzielle Risiko von Anwendern und Softwareentwicklern bei Investitionen in Linux verringert.

Interessante Eigenschaften Das Linux-Betriebssystem verfügt über eine Reihe äußerst interessanter Eigenschaften für IT-Manager im privaten und öffentlichen Bereich: So sind für Anwender, die auf Linux umstellen, Lock-in-Effekte weniger wahrscheinlich als bei der Nutzung von proprietären Software-Plattformen. Darüber hinaus haben sie größere Einflussmöglichkeiten auf ihre Computerumgebung. Diese Vorteile kommen ganz besonders in komplexen Computerumgebungen zur Geltung, wobei Anwender von der Fähigkeit profitieren, ihre Softwareumgebung an individuelle Bedürfnisse anpassen zu können, was gerade im öffentlichen Sektor häufig erforderlich ist.

Offene/freie Schnittstellen Charakteristisch für Open-Source-Software wie Linux ist die Nutzung offener Schnittstellen. Proprietäre Software verwendet sowohl offene als auch proprietäre Schnittstellen. Da offene Schnittstellen in der Softwareindustrie zu mehr Stabilität und Innovation führen, sollte ihnen und der entsprechenden Software von Mitarbeitern im öffentlichen Dienst der Vorzug gegeben werden, sofern die gebotene Qualität mit der von proprietären Alternativen vergleichbar ist.

Vorteile für lokale SME-Server[4] Da Linux eine Open-Source-Software-Plattform ist, kann dessen Einführung dabei helfen, die Entwicklung des landeseigenen Softwaremarktes anzuregen, teilweise indem die Ausbildung von Programmierern gefördert wird. Diese befähigt sie zur Entwicklung von Anwendungen, die auf der Linux-Plattform ausführbar sind. Die Einführung von Linux kann sich darüber hinaus günstig auf die Entwicklung von proprietärer Software und Dienstleistungen auswirken, die in der Linux-Umgebung genutzt werden können.

Das Beste aus beiden Bereichen Dass sich die mit Linux verknüpften Lizenzbedingungen negativ auf die Entwicklung kommerzieller Software auswirken könnten, ist eine unbegründete Befürchtung. Auch wenn Linux Open-Source-Software ist, bedeutet das nicht zwangsläufig, dass die Entwicklung von Anwendersoftware, die unter Linux ausgeführt werden kann, ebenfalls dem Open-Source-Modell folgen muss. Eher ist die Entstehung von gemischten Umgebungen zu erwarten, die sowohl Open-Source-Software als auch proprietäre Software nutzen und über offene und proprietäre Schnittstellen verfügen, die in ihrer Gesamtheit den wirtschaftlichen Erfolg über Jahre hinaus garantieren können.

4 *Small and medium size enterprises* (SME).

3 Besprechung von proprietärer und Open-Source-Software

GNU/Linux ist derzeit das führende Beispiel für OSS.[5] Wer die Einführung von Linux in Betracht zieht, muss ein Verständnis dafür entwickeln, wie das Open-Source-Modell für Softwareentwicklung funktioniert und inwiefern OSS eine Ergänzung zu proprietärer Software darstellen kann. Zur Verbesserung dieses Verständnisses werden im vorliegenden Artikel OSS, proprietäre Software und Schnittstellen zunächst allgemein besprochen. Diese Informationen dienen später als Grundlage, wenn wir Kosten und Nutzen bei der Einführung von Linux bewerten.

3.1 Open-Source-Software und proprietäre Software

Unter OSS versteht man Software mit öffentlich zugänglichem Quelltext[6], der von jeder Person kopiert, verändert und weitergegeben werden kann. Der Zugang zum Quelltext bietet Programmierern und Anwendern die Möglichkeit, das zu Grunde liegende Programm einzusehen und zu verstehen. Sie können den Quelltext sogar erweitern oder verändern, wobei bestimmte Lizenzbeschränkungen bestehen können, auf die später noch eingegangen werden soll.

Im Gegensatz dazu versteht man unter proprietärer Software solche, die im Zusammenhang mit proprietären Lizenzvereinbarungen vertrieben wird, normalerweise unter Zahlung einer Lizenzgebühr. Obwohl es viele verschiedene Formen der proprietären Softwarelizenzierung gibt, erhalten Anwender von proprietärer Software sehr häufig keinen Zugang zu dem urheberrechtlich geschützten Quelltext und dürfen für gewöhnlich weder die Software selbst noch Erweiterungen zur Software weitergeben. Unternehmen, die proprietäre Software entwickeln, sichern sich in der Regel die geistigen Eigentumsrechte, um die alleinige Kontrolle über den von ihnen entwickelten Quelltext zu behalten.

3.2 Offene und proprietäre Schnittstellen

Es gibt praktisch keine Software, die vollkommen unabhängig arbeitet. Deshalb stellen Schnittstellen, d. h. die Art und Weise, wie ein Programm mit anderen Programmen, dem Anwender und mit der Hardware interagiert, für den Anwender und die Funktionalität entscheidende Komponenten dar. Schnittstellen sind besonders wichtig in komplexen Computerumgebungen, die in Regierungsbehörden und großen kommer-

5 Die allgemein als Linux bezeichnete Betriebssystem-Software sollte richtiger *GNU/Linux* genannt werden, da es sich dabei um eine Kombination von Software (GNU-Projekt) der *Free Software Foundation* und einem von Linus Torvalds entwickelten *Kernel* handelt. GNU ist ein Akronym für *GNU's not Unix*.

6 Programmierer schreiben Software in verschiedenen Computersprachen wie beispielsweise *Fortran*, *C/C++* und *Java*. Das ursprüngliche Format, in dem die Software entwickelt wurde, wird *Quelltext* genannt. Dieser Quelltext wird in ein ausführbares, binäres oder Zieltextformat übersetzt, mit dem der Computer arbeiten kann.

ziellen Einrichtungen charakteristisch sind. Es lassen sich drei verschiedene Arten von Schnittstellen unterscheiden:

Programmierschnittstellen, die allgemein als *APIs* bezeichnet werden und durch die beschrieben wird, inwiefern Anwendungen bestimmte Dienste vom Betriebssystem erfordern.

Benutzerschnittstellen, durch die beschrieben wird, wie der Benutzer Software visuell wahrnimmt und mit ihr interagiert.

Dokumentenformate, durch die beschrieben wird, wie Daten von den Anwendungen interpretiert und abgespeichert werden.

Es kann weiterhin zwischen offenen und proprietären Schnittstellen unterschieden werden. Eine Schnittstelle, die von einer einzelnen Gruppe kontrolliert wird und nicht jedem zur freien Nutzung zur Verfügung steht, wird proprietäre Schnittstelle genannt. Viele kommerzielle Softwareunternehmen halten an proprietären Schnittstellen fest, wofür Dateiformate ein gängiges Beispiel sind. Auf der anderen Seite gibt es offene Schnittstellen[7], die in öffentlich zugänglichen Dokumenten vollständig dokumentiert werden und jedem zur freien Verfügung stehen. Das *Open Document Format (ODT)* beispielsweise ist ein Standardformat für editierbare Office-Dokumente, das vom OASIS-Konsortium vollständig beschrieben wurde und öffentlich zugänglich ist, anstatt sich unter alleiniger Kontrolle eines einzigen Unternehmens zu befinden – daher handelt es sich um eine offene Schnittstelle.[8] In Zusammenhang mit OSS kommen in aller Regel offene Schnittstellen vor, die im Quelltext vollständig dokumentiert werden.

Offene Schnittstellen bieten Anwendern und Softwareentwicklern zahlreiche Vorteile. Sie erweitern das Angebot für die Verbraucher und fördern dadurch den Wettbewerb. Darüber hinaus erhöhen sie die Kompatibilität einer Vielzahl von Programmen untereinander, auch wenn diese im Laufe der Zeit im Zuge von Verbesserungen verändert werden. Wie ausführlich in unserem Buch (Shapiro und Varian 1999) diskutiert wird, können daher sowohl Softwarehändler als auch Anwender von der Einrichtung offener Schnittstellen profitieren, was zusätzlich zur Vermeidung von Konflikten um Standards zwischen inkompatiblen proprietären Schnittstellen beiträgt.

7 Eine offene Schnittstelle ist dokumentiert und steht jedem, ohne Einschränkungen durch den Autor, für die freie Nutzung zur Verfügung. Es gibt einige Grauzonen, z. B. veröffentlichte Schnittstellen, die durch einzelne Gruppen kontrolliert werden; veröffentlichte Schnittstellen mit Nutzungseinschränkungen; Schnittstellen, die innerhalb einer bestimmten Gruppe geteilt werden, aber nicht öffentlich zugänglich sind; multiple Schnittstellen usw. Die unten beschriebenen allgemeinen Fälle reichen jedoch für unsere Zwecke aus.

8 Siehe http://www.oasis-open.org.

3.3 Open-Source-Software und proprietäre Software als sich ergänzende Komponenten

Die Verdeutlichung der grundsätzlichen Unterschiede zwischen OSS und proprietärer Software als Modelle zur Softwareentwicklung soll nicht zu dem Schluss führen, dass sich Anwender zwischen der einen oder anderen Softwareart entscheiden müssen, um all ihre Bedürfnisse im Hinblick auf Computerarbeit zu erfüllen. Wir sind sogar im Gegenteil der Überzeugung, dass OSS und proprietäre Software zukünftig nebeneinander existieren und einander ergänzen können und werden. Schon jetzt ist eine Reihe von proprietären Anwendungen mit Linux ausführbar, darunter beispielsweise die *WebSphere* von *IBM* und Datenbankprogramme von *Oracle* und *IBM*. In gleicher Weise arbeitet OSS, einschließlich eines großen Teils der GNU-Software, mit den Betriebssystemen *Solaris* von *Sun* und Windows von *Microsoft*.

Aus genau diesen Gründen werden öffentliche Institutionen durch die Einführung von Linux in keiner Weise daran gehindert, für viele Anwendungen proprietäre Software einzusetzen, die unter Linux ausführbar ist. Tatsächlich stellen die Zugänglichkeit des Linux-Quelltexts und die offenen Schnittstellen zwischen Linux und Anwendungssoftware sogar große Vorteile für die Entwicklung kommerzieller Softwareanwendungen dar, die unter Linux ausgeführt werden können.

3.4 Für Open-Source-Software förderliche Wirtschaftsbedingungen

Weltweit werden die Anwender von dem zukünftigen Wettbewerb zwischen OSS und proprietärer Software profitieren können. Man kann sicherlich davon ausgehen, dass jeder der unterschiedlichen Softwaretypen in bestimmten Bereichen besonders erfolgreich sein wird, was zur Förderung des Wettbewerbs im Softwaresektor beiträgt und deshalb unserer Meinung nach positiv zu bewerten ist. Insbesondere Linux kann von einer Reihe vorteilhafter Marktbedingungen profitieren:

Nachweisbare Erfolge Es existieren zahlreiche Nachweise für den Erfolg von Linux als kostengünstige Variante beim Einsatz in großen, verlässlichen Computersystemen.

Flexibilität und Einflussnahme Viele Anwender legen großen Wert auf die hohe Flexibilität und den Erhalt weitreichender Einflussnahmemöglichkeiten auf ihre Server-Betriebssysteme, die ihnen Linux im Vergleich zu proprietärer Software bietet.

Weitverbreitete Anwendung Linux wird bereits von einer großen Zahl von Anwendern genutzt, wodurch sich einerseits das Angebot von Anwendungen erweitert, die unter Linux ausführbar sind, und andererseits immer mehr Programmierer zur Verfügung stehen, die sich mit Linux auskennen.

Starke Gemeinschaft Linux kann auf die Fähigkeiten einer vielfältigen und starken Entwicklergemeinschaft zurückgreifen, deren hohe Motivation aus der Tatsache resultiert, dass die Beteiligung am Linux-Projekt Status und Anerkennung verspricht.

Governance-Strukturen Linux verfügt über die notwendigen Führungsstrukturen und Institutionen, um einer Zersplitterung der Linux-Gemeinschaft vorzubeugen und einen Leitplan für die weitere Entwicklung zu erstellen.

Unterstützung durch Wirtschaftsunternehmen Linux wird von führenden Technologieunternehmen unterstützt, für die das Angebot eines auf die Anwenderbedürfnisse zugeschnittenen integrierten Paketes und/oder von Ergänzungen zum Linux–Betriebssystem wirtschaftlich interessant ist. Zu den wichtigsten Beispielen zählen Unternehmensplattformen und -dienstleistungen von *Red Hat*, Prozessoren von *Intel* oder Server und damit verbundene Dienstleistungen von *IBM*.

4 Einführung von Linux: Nutzen und Kosten für die Anwender

Auf Grundlage der bisherigen Überlegungen können nun Kosten und Nutzen systematisch erörtert werden, die für Anwender mit der Einführung des Linux-Betriebssystems als Plattform, auf der sie ihre Computerumgebung einrichten, verbunden sind. Wie schon angedeutet, wird Linux bereits von vielen großen Organisationen im privaten und öffentlichen Bereich genutzt. Im Folgenden geht es um Informationen, die bei der Entscheidung helfen können, ob der Einsatz von Linux im Einzelfall sinnvoll ist oder nicht.

4.1 Total Cost of Ownership

Fachleute und Berater stimmen darin überein, dass die *Total Cost of Ownership* (TCO) einen wichtigen Gesichtspunkt bei der weitreichenden Entscheidung über die Einführung neuer Software darstellen sollten. Obwohl die TCO in der Theorie genau definiert sind, kann ihre Bestimmung im Einzelfall eine äußerst komplexe Angelegenheit sein. Zwar können die Anschaffungskosten selbst problemlos angegeben werden, sie machen aber nur einen Teil der TCO aus. Hinzu kommen Ausgaben für Anwenderschulungen, Wartungsarbeiten, Upgrades und fachliche Unterstützung, die im Vergleich zum Kaufpreis sogar einen größeren Anteil der TCO darstellen können. Da viele dieser Kosten erst nach dem eigentlichen Softwareerwerb anfallen, sind Vergleiche zudem nur dann aussagekräftig, wenn durch sie die Ausgaben für den gesamten Zeitraum des Softwareeinsatzes berücksichtigt werden. Häufig muss sogar über die zeitliche Dauer des Projekts hinausgeblickt werden, für das die Entscheidung

zur Einführung neuer Software getroffen wurde. Denn in vielen Fällen können Daten, Schulungen und Arbeitsabläufe das Arbeitsprojekt überdauern, mit dem sie in Zusammenhang stehen.

Es hat unterschiedliche Versuche gegeben, die TCO von Windows und Linux in verschiedenen Computerumgebungen miteinander zu vergleichen.[9] Der Unterschied bewegt sich meistens in Größenordnungen zwischen 10 und 15 Prozent und ist damit eher unbedeutend, denn solche Differenzen bei den TCO können leicht durch spezifische regionale Bedingungen, zufällige Ereignisse o. Ä. verwischt werden. Daher scheint auf den ersten Blick die Annahme begründet zu sein, dass bei einer konventionellen TCO-Analyse keine der beiden Plattformen über einen nennenswerten Vorteil gegenüber der anderen verfügt. Einen sehr großen Anteil der TCO machen die Lohnkosten für Systemadministratoren und Servicepersonal aus. Dabei wurden in den oben erwähnten Studien grundsätzlich die marktüblichen Löhne in den USA zu Grunde gelegt. Diese Kosten können jedoch in Ländern mit niedrigeren Lohnkosten wesentlich geringer sein. Da schon der Anschaffungspreis von Linux deutlich niedriger ist als für proprietäre Alternativen, würden die TCO, welche neben dem Kaufpreis für die Software auch die für ihren Einsatz erforderlichen Lohnkosten enthalten, in solchen Ländern wesentlich geringer ausfallen.

4.2 Umstellungskosten

Es ist ratsam, vor der Entscheidung für die Einführung neuer Software auch den Umstellungskosten besondere Beachtung zu schenken. Neben den bei der Umstellung auf das neue System direkt anfallenden Kosten sind auch die Ausgaben bei einer späteren Umstellung von diesem System zu einem anderen zu bedenken. Vorsicht ist geboten bei der Einführung eines Systems, das einen zukünftigen Wechsel schwierig machen könnte, beispielsweise durch Lock-In-Effekte, die bei der Nutzung des Systems auftreten und die zukünftige Verhandlungsposition dem Anbieter gegenüber schwächen.

Während Systemanbieter in gewisser Weise immer bestrebt sind, den Anwendern die Umstellung auf Alternativen zu erschweren, wollen diese normalerweise ihre Flexibilität erhalten. Aus der Sicht der Anwender ist es besonders wichtig, sicherzustellen, dass Dateiformate, Daten, Systemaufrufe, Programmierschnittstellen (*APIs*) und andere Schnittstellen, Kommunikationsstandards und Ähnliches ausreichend dokumentiert sind, um Daten und Programme leicht von einem Anbieter zum anderen übertragen zu können. In dieser Hinsicht bietet OSS mit ihren offenen Schnittstellen dem Anwender einen klaren Vorteil gegenüber proprietärer Software mit proprietären Schnittstellen. Natürlich kann proprietäre Software diesen Vorteil neutralisieren, wenn sie selbst über echte offene Schnittstellen verfügt. In diesem Zusammenhang

9 Siehe die zwei Studien, die auf Anfrage der Robert Francis Group (2002) und Cybersource (2002) durchgeführt wurden. Siehe auch die Studie von Bozman et al. (2002), die von *Microsoft* in Auftrag gegeben wurde.

wirkt sich die Unterstützung von OSS im Allgemeinen und Linux im Speziellen dadurch positiv aus, dass Druck auf Anbieter proprietärer Software ausgeübt wird, ihre Schnittstellen zum Vorteil der Anwender zu öffnen.

Eine weitere wichtige Feststellung ist das Bestreben der Marktführer einer Branche, die alleinige Kontrolle über ihre Schnittstellen auszuüben und anderen Anbietern Interoperabilität zu erschweren. Umgekehrt haben aber gerade Marktakteure, die sich nicht in der dominanten Position befinden, eine sehr starke Motivation auf diesem Gebiet mit dem Marktführer und untereinander zu interoperieren.

4.3 Softwarequalität

Die Softwarequalität lässt sich mit Hilfe verschiedener Kriterien bestimmen. Wichtig sind vor allem Verlässlichkeit, Wartbarkeit, Benutzerfreundlichkeit, Sicherheit und Flexibilität. Wer die Einführung neuer Software in Erwägung zieht, muss vor der Wahl eines Systems zunächst klären, welche Rolle die jeweiligen Kriterien in der eigenen Umgebung spielen. Am einfachsten lässt sich das Verhältnis zwischen den einzelnen Qualitätskriterien und den verschiedenen Modellen zur Softwareentwicklung erklären, indem die jeweiligen Vorzüge von Windows und Linux beschrieben werden. Allerdings muss darauf hingewiesen werden, dass diese beiden Systeme kaum „repräsentativ" sind. Es handelt sich dabei um zwei der besten derzeit erhältlichen Softwarepakete, die eine zentrale und bestimmende Rolle in Computerumgebungen spielen. Von Linux kann jedoch ebenso wenig direkt auf andere Open-Source-Projekte geschlossen werden wie von Windows auf andere Softwaresysteme mit nicht zugänglichem Quelltext. Der Vergleich anderer Vertreter für OSS und proprietäre Software kann zu ganz anderen Ergebnissen führen. Dennoch können bestimmte Vergleiche die unterschiedlichen Ansätze der Entwicklungsmodelle aufzeigen.

Verlässlichkeit

Die Verlässlichkeit bestimmter OSS-Programme wird seit langem anerkannt. Die Tatsache, dass Linux große Webseiten wie *Google* betreibt, stellt seinen Ruf als verlässliche Software unter Beweis. Obwohl sich die Verlässlichkeit von Windows in den neueren Versionen erheblich verbessert hat, ist es dennoch fraglich, ob dasselbe Niveau wie bei Linux erreicht wird. Im FLOSS-Report, der durch Befragungen in 1452 europäischen Organisationen entstand, gaben 83 Prozent der Teilnehmer an, dass „größere Stabilität" ein wichtiger oder sehr wichtiger Grund für die Einführung von OSS war (Gosh et al. 2002). Auch wenn die neueren Betriebssysteme von *Microsoft* den Ruf haben, verlässlicher zu sein als frühere Versionen, scheint ihre Architektur im Vergleich zu Linux einen inhärenten Nachteil aufzuweisen: das Fehlen eines umfassenden modularen Aufbaus. Es existiert sogar die Auffassung, dass die Verlässlichkeit des Linux-Systems eine Nebenerscheinung seines modularen Aufbaus ist. Demnach ist es einfacher, Bestandteile eines modular aufgebauten Softwaresystems zu ersetzen, ohne

die Arbeit anderer Teile zu beeinträchtigen, als bei holistischen, integrierten Systemen. Diese für das Open-Source-Entwicklungsmodell charakteristische Modularität ist geradezu eine Voraussetzung für Systeme, die von mehreren, mehr oder weniger unabhängig voneinander arbeitenden Programmierern entwickelt werden.

Wartbarkeit

Die Wartbarkeit bezieht sich auf den Aufwand im Zusammenhang mit der Aktualisierung und Erhaltung der Betriebsbereitschaft eines Systems. In der Vergangenheit konnten Paketaktualisierungen unter Linux leichter durchgeführt werden als unter Windows, weil Linux standardisierte Paketmanagement-Verfahren verwendet, um die Komplexität beherrschen zu können. Alle Dateien eines bestimmten Programms werden grundsätzlich an wenigen, genau dokumentierten Orten abgespeichert. Die meisten Konfigurationsprogramme sind textbasiert, so dass sie von den Anwendern problemlos konsultiert werden können. Auch hier wurden Berichten zufolge in den neuesten Versionen von Windows Verbesserungen im Hinblick auf die Wartbarkeit des Systems erreicht, so dass Patches und Updates fast automatisch durchgeführt werden können. Dennoch gaben im FLOSS-Report 60 Prozent der befragten Teilnehmer an, dass Einsparungen bei den Betriebs- und Administrationskosten ein wichtiger oder sehr wichtiger Grund für den Einsatz von Linux seien (Gosh et al. 2002).

Benutzerfreundlichkeit

Die Prüfung auf Benutzerfreundlichkeit ist mit hohen Kosten verbunden. Proprietäre Plattformen für Einzelanwender halten in dieser Beziehung der Open-Source-Konkurrenz gegenüber einen langjährigen Vorsprung, insbesondere bei der Benutzeroberfläche. Vor allem *Microsoft* hat in den letzten Jahren stark in die Benutzerfreundlichkeit seiner Anwendungen investiert. Dennoch haben auch einige viel beachtete Open-Source-Projekte wie *GNOME* und *KDE*, die sich speziell mit Benutzeroberflächen beschäftigen, in den letzten Jahren bedeutende Fortschritte gemacht.

Hinzu kommt, dass mit Linux eine umfassende Anpassung der Computerumgebung an individuelle Bedürfnisse möglich ist, wodurch jeder Benutzer mit seiner eigenen Umgebung arbeiten kann. Unter anderem können die gängigen Linux-Anwendungen so konfiguriert werden, dass sie wie unter Windows aussehen und arbeiten, wodurch den Anwendern der Systemwechsel zusätzlich erleichtert wird. Da die Technologien für graphische Benutzerschnittstellen immer mehr ausreifen, werden „revolutionäre" neue Features immer seltener und stellen eher Zwischenschritte dar, wodurch Konkurrenten mehr Zeit haben, um auf bedeutsame Vorstöße zu reagieren. Dadurch ist zu erwarten, dass sich die Benutzerfreundlichkeit von Linux-Software zunehmend verbessern wird (Nichols und Twidale 2003).

Sicherheit

Es gibt eine anhaltende Diskussion über die Sicherheit von Windows und Linux. Da Linux als Mehrbenutzer-System angelegt ist, verfügt es über viele integrierte Schutzmechanismen zur Gewährleistung der Benutzersicherheit. Da der Quelltext des Systems frei verfügbar ist und von vielen Entwicklern auch genutzt wird, ist das Aufspüren von Programmfehlern (Bugs) wesentlich einfacher. Allerdings trifft das für beide Seiten zu: Angreifern wird es ebenso leicht gemacht, Bugs zu finden, wie Verteidigern, sie zu beseitigen. Wurde ein Programmfehler einmal aufgespürt, ist die Behebung des Problems mit OSS einfacher, weil jede Person die Quelle kontrollieren und den Bug analysieren kann. Darüber hinaus können versierte Anwender ihre Systeme selbst konfigurieren, um die gängigen Schwachpunkte zu beseitigen. Das Betriebssystem *Security-Enhanced Linux* von der *NSA* beispielsweise verzichtet an manchen Stellen auf Funktionalität zugunsten einer erhöhten Sicherheit. Die Offenlegung des Quelltexts von sicheren Systemen bietet potenziellen Anwendern daher eine gewisse Absicherung gegen etwaige Sicherheitslücken.[10]

Flexibilität

OSS verfügt über einen hohen Grad an Flexibilität, da sie individualisiert und an spezifische Bedürfnisse angepasst werden kann. Diese Eigenschaften haben die Experimentierfreudigkeit und die Arbeit an Verbesserungen gefördert, was zu einer beachtlichen Menge an Innovationen von Seiten der Anwender geführt hat (von Hippel 2002). Dadurch konnte sich Linux in vielerlei Hinsicht in ganz unvorhergesehene Richtungen entwickeln.[11] Insbesondere den folgenden Aspekten wurde bei den zahlreichen Entwicklungs- und Individualisierungsbemühungen Aufmerksamkeit geschenkt:

Klein Eine Art der individuellen Anpassung besteht in der Eliminierung aller Linux-Bestandteile, die für eine genau eingegrenzte, spezifische Aufgabe nicht gebraucht werden. So passt *Mindi Linux*, das für die Datenwiederherstellung eingesetzt werden kann, auf eine einzelne Diskette. Andere Ein-Disketten-Ausführungen von Linux sind *Coyote Linux*, *Trinux* und das *Linux Router Project*, die ebenfalls für diverse Anwendungen, wie beispielsweise Netzwerkaufgaben,

10 Die Nutzer von Open-Source-Software sind davon überzeugt, dass sie über bessere Sicherheitseigenschaften verfügt. Im FLOSS-Report gaben 75 Prozent der Befragten „besseren Zugangsschutz" als wichtigen oder sehr wichtigen Grund für die Einführung von Open-Source-Software an (Gosh et al. 2002). Der bekannte Sicherheitsexperte Ross Anderson greift auf ein theoretisches Modell der Softwarequalität bei seiner Argumentation zurück, dass offene und geschlossene Softwaresysteme weitestgehend denselben Grad an Verlässlichkeit aufweisen (Anderson 2002).

11 Eine eindrucksvolle Auflistung der enormen Vielfalt an Linux-Distributionen ist unter http://lwn.net/Distributions zu finden. Die Liste enthält Distributionen für spezifische Hardware-Anforderungen, für sehbehinderte Anwender, für *ISP*s, für Echtzeitanwendungen (wie Einrichtungsüberwachung), für Multimedia und eine große Zahl anderer Bedürfnisse.

optimiert wurden. Die verschiedenen Ausführungen des eingebetteten Linux–Systems[12] können für bestimmte Hardwareaufgaben eingesetzt werden, etwa als Registrierkassen, *PDA*s, tragbare Videorecorder (z. B. *Tivo*), MP3-Player und Ähnliches.

Größer und Leistungsfähiger Mit Hilfe von Linux können Computer zu Clustern verbunden werden, die leistungsfähigere Rechnersysteme bilden und vielfältige Aufgaben übernehmen können, z. B. die Auswertung größerer Datenmengen, die zentrale Speicherung von Daten auf Dateiservern, die Bereitstellung von Datenbanken und den Betrieb von Webservern, aber auch Flugsimulationen, die Wiedergabe von Computergraphiken oder Wettersimulationsmodelle. In letzter Zeit gibt es bei Linux auch Bemühungen zur Einführung von *grid computing*, was Institutionen die Nutzung der Rechenleistung umfangreicher *computer arrays* erlaubt, die über das Internet bereitgestellt wird.[13]

Hochsicher Die bereits erwähnte Realisierung eines sicheren Linux-Betriebssystems der *NSA* ist nur eins von zahlreichen Projekten zur „Abhärtung" von Linux. Andere Beispiele sind u. a. *Engarde Linux* und *Bastille Linux*.

Lokal angepasst Es ist kein Zufall, dass Open Source in kleineren Sprachgemeinschaften besonders beliebt ist. Ein häufiger Wunsch bei den Nutzern sind Anwendungen, mit denen sie in ihrer Muttersprache arbeiten können. Wo solche Anwendungen im derzeitigen proprietären Softwareangebot nicht existieren, haben die Nutzer teilweise ihre eigenen Open-Source-Versionen geschrieben.[14]

Freiheiten bei Anpassungen und bei firmeneigenen Entwicklungen werden durch Linux in einem Ausmaß ermöglicht, das mit proprietärer Software unmöglich zu erreichen ist. Daher sind Manager, die Flexibilität und Kontrolle anstreben, gut beraten, sich für Linux zu entscheiden, um so das Fachwissen in der Firma aufzubauen, welches zur Ausnutzung der Vorteile von Open Source nötig ist.

5 Open Source in Wirtschaft und Bildung

Wer sich in öffentlichen Institutionen mit der Wahl einer Software-Plattform befasst, ist verpflichtet, das System zu wählen, das für die anstehende Aufgabe am besten geeignet ist. Die bisherigen Betrachtungen konnten zeigen, dass Linux im Vergleich zu proprietären Alternativen in vielen Fällen über gleichwertige oder bessere Eigenschaften in Bezug auf Leistung, Verlässlichkeit und Sicherheit verfügt. Für den Fall,

12 Für mehr Informationen zu eingebetteten Linux-Systemen siehe auch http://www.linuxdevices.com.

13 Nähere Einzelheiten beim *Linux Clustering Information Center* unter http://lcic.org.

14 Für Teilauflistungen von lokal angepassten Ausführungen von Linux siehe auch http://www.linux.org/docs/ldp/howto/HOWTO-INDEX/other-lang.html und http://www.linuxselfhelp.com/cats/localization_language.html.

dass zwei Systeme gleich gut für eine bestimmte Aufgabe geeignet sind, werden offene Schnittstellen zu einem entscheidenden Aspekt, da sie üblicherweise mit niedrigeren Kosten für Zusammenschaltung und Umstellung verbunden sind. Dadurch verringert sich die Wahrscheinlichkeit, dass der Kunde an einen einzelnen Anbieter gebunden wird (*lock-in*). Offene Schnittstellen regen unabhängige Programmierer zur Entwicklung von Anwendungen, *add-ons* und Ergänzungsprodukten an. Der Nutzen solcher Produkte ist nicht nur für Anwender, sondern auch auf nationaler Ebene so groß, dass praktisch jeder Softwareanbieter Anspruch auf Offenheit erhebt. Die entscheidende Frage ist jedoch, ob die Anbieter tatsächlich bestrebt sind, diesem Anspruch in der Praxis auch gerecht zu werden, und zwar nicht nur vorläufig, sondern auch zukünftig, wenn die Umstellung auf ein anderes System sehr hohe Kosten verursachen kann.

Länder, die an einer starken einheimischen Softwareindustrie interessiert sind, sollten sich zunächst mit ihrem Universitätssystem befassen und sich die Frage stellen, wie die Ausbildung ihrer zukünftigen Softwareentwickler gefördert werden kann. Da durch offene Schnittstellen die lokale Entwicklung von unabhängigen Anwendungen ermöglicht wird, spielen sie hierbei eine entscheidende Rolle. Die entsprechenden Möglichkeiten werden einheimischen Softwareunternehmen nicht nur von Open-Source-Software-Plattformen selbst geboten, sondern auch von proprietären Software-Plattformen mit offenen Schnittstellen. Im Gegensatz dazu stellen proprietäre Software-Plattformen mit proprietären Schnittstellen für unabhängige Entwickler einen strategischen Nachteil den Unternehmen gegenüber dar, die die Schnittstellen zwischen Software-Plattform und Anwendungen kontrollieren. Eine solche strategische Abhängigkeit kann bei der Entwicklung proprietärer Anwendungssoftware vor Kapital- und Personalinvestitionen abschrecken.

Open Source spielt auch deshalb eine wichtige Rolle, weil dadurch die internen Arbeitsprozesse der Software sichtbar gemacht werden, wodurch Studenten genau verfolgen können, wie qualitativ hochwertige Software entsteht. Genau wie angehende Automechaniker während ihrer Ausbildung mit echten Motoren arbeiten, müssen sich angehende Systemprogrammierer an realen Betriebssystemen ausprobieren können. Die Verfügbarkeit von Systemen, die durch freien Zugang ein effektives Studium ermöglichen, wird einen großen Beitrag bei der Ausbildung fähiger Informatiker und der Entwicklung hochwertiger Produkte leisten.

6 Schlussfolgerung

Open-Source-Software ist zukünftig nicht mehr wegzudenken. Was einmal als völlig neuer, geradezu häretischer Ansatz für Softwareentwicklung galt, hat sich inzwischen in der Praxis hinreichend bewährt. Es existieren realisierbare Geschäftsmodelle für OSS-Entwickler, und jeder Anwender wird aus dem wohl überlegten Einsatz von OSS als Ergänzung zu proprietärer Software Nutzen ziehen können. Insbesondere Linux ist zum Paradebeispiel eines erfolgreichen Open-Source-Projekts gereift, weil

durch den Aufbau ständiger Institutionen der Linux-Quelltext permanent verbessert werden kann. Viele Nutzer im privaten und öffentlichen Bereich werden sicherlich stark von der Einführung von Linux in ihrer Computerumgebung profitieren. Linux hat gezeigt, dass es auch in sehr anspruchsvollen Computerumgebungen erfolgreich eingesetzt werden kann und dabei eine Reihe entscheidender Vorteile bietet. Dazu zählen Verlässlichkeit, Flexibilität, Sicherheit und die Vermeidung der Lock-In-Effekte von proprietären Alternativen.

Für Technologiemanager im öffentlichen Sektor bestehen noch weitere Gründe für die Einführung von Linux: Das Linux-Betriebssystem fördert die Ausbildung von Programmierern und stellt eine offene Plattform dar, mit deren Hilfe proprietäre und Open-Source-Anwendungen entwickelt werden können, was auf nationaler Ebene zur Stärkung der Softwareindustrie führt. Aus all diesen Gründen sollten IT-Manager, die im öffentlichen Bereich flexible Computerumgebungen aufbauen und gleichzeitig die einheimische Softwareindustrie fördern wollen, den Einsatz von Linux und von Open-Source-Software allgemein unbedingt in Erwägung ziehen.

Literatur

Anderson, R. (2002), Security in Open and Closed Systems: The Dance of Boltzman, Coase, and Moore, Working Paper, Cambridge University. http://www.cl.cam.ac.uk/ftp/users/rja14/toulouse.pdf [27. Jan 2007].

Bozman, J., Gillen, A., Kolodgy, C., Kusnetzky, D., Perry, R. und Shiang, D. (2002), Windows 2000 Versus Linux in Enterprise Computing, IDC White Paper, IDC. http://www.microsoft.com/windows2000/docs/TCO.pdf [27. Jan 2007].

Cybersource (2002), Linux vs. Windows: Total Cost of Ownership Comparison, Study, Cybersource. http://www.cyber.com.au/cyber/about/linux_vs_windows_tco_comparison.pdf [27. Jan 2007].

Gosh, R., Krieger, B., Glott, R. und Robles, G. (2002), FLOSS – Free/Libre and Open Source Software, Survey and Study commissioned by the EU, International Institute of Infonomics, University of Maastricht und Berlecon Research GmbH. http://www.infonomics.nl/FLOSS/report/ [27. Jan 2007].

Nichols, D. M. und Twidale, M. B. (2003), 'The Usability of Open Source Software', *First Monday* **8**(1). www.firstmonday.org/issues/issue8_1/nichols/ [27. Jan 2007].

Robert Francis Group (2002), Total Cost of Ownership for Linux in the Enterprise, Study, Robert Francis Group. http://www.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf [27. Jan 2007].

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

von Hippel, E. (2002), Open Source Software Projects as User Innovation Networks, Working Paper, MIT Sloan School of Management. http://idei.fr/doc/conf/sic/papers_2002/vonhippel.pdf [27. Jan 2007].

Linux fensterlt in der Verwaltung

STEFAN KREMPL*

(CC-Lizenz siehe Seite 563)

Der öffentliche Sektor entwickelt sich zum Treiber für Open-Source-Projekte, doch Microsoft hält mit konstanten Lobby-Bemühungen dagegen: Die Stadt München und das Auswärtige Amt sind die Eisbrecher im Verwaltungssektor beim Abdocken von Microsoft. Sie gelten als anschauliche Beispiele dafür, dass es ernsthafte Alternativen zu den von Redmond vorgegebenen Update-Zyklen gibt, dass sich Lizenzkosten sparen und klamme Haushaltskassen entlasten lassen. Die Redmonder sind aber immer wieder für eine Überraschung im Kampf gegen den ungeliebten Wettbewerber Linux gut. So versuchen sie etwa, die Befürworter von Open Source mal übers Vergabe- und mal übers Patentrecht stolpern zu lassen.

Schlüsselwörter: Öffentliche Verwaltung · Lobbying · Auswärtiges Amt · Microsoft · LiMux

1 Einleitung

Es dauert ein Weilchen, bis das System an dem Rechner im EDV-Amt der Stadt München unweit des Stachus sich auf den fremden Benutzer eingestellt und die richtigen Konfigurationen gefunden hat. Doch dann taucht der Begrüßungsbildschirm mit einem Panoramabild der Altstadt mit Blick auf das Rathaus sowie Theatiner- und Frauenkirche, einem Pinguin mit jahreszeitlich passendem Snowboard und dem Hinweis „Der LiMux-Basisclient wird gestartet“ über einem Balken auf, der den Stand des Bootvorganges wiedergibt. Mit zwei weiteren Mausklicks öffnet Florian Schießl vom LiMux-Projektbüro der bayerischen Landeshauptstadt das Textverarbeitungsprogramm *OpenOffice.org* und zaubert einen Briefkopf auf den Schirm, der auch an dem von ihm normalerweise nicht genutzten Arbeitsplatz alle seine persönlichen Adressdaten bis hin zu den korrekten Verkehrsanschlüssen für seinen Amtssitz enthält.

* Der Autor dankt Bastian Zimmermann für die wertvolle Vorarbeit zu diesem Artikel. Auf ihn gehen die ursprüngliche Initiative für diesen Text und die Kontakte zu vielen Interviewpartnern zurück.

Anfang des Jahres 2007 läuft Linux nebst freier Software für die Internetnutzung und einer Open-Source-Office-Suite in einer eigens erstellten Client-Version auf rund 400 Desktops in München im produktiven Einsatz. Das Revisionsamt der Stadt zog im Herbst 2006 vollständig auf die Linux-Variante *LiMux* um. Auch die komplette Stadtspitze samt Oberbürgermeister Christian Ude und Bürgermeisterin Christine Strobl gehört zu den Pionieren, deren E-Mails am Ende ein dezenter Hinweis ziert: „Wurde von einem LiMux-Arbeitsplatz aus versandt." Bis zum Jahresende sollen es 1 500 Rechner sein, auf denen freie Software Windows und *Microsoft Office* ablöst. Danach sollen nach und nach bis Mitte 2009 gut 80 Prozent der etwa 14 000 PCs in den Behörden der Hauptstadt mit Herz auf *LiMux* umgerüstet werden. Den Schwerpunkt sieht der EDV-Chef Münchens, Wilhelm Hoegner, dabei zunächst noch auf der Office-Migration auf Basis des vorhandenen *Windows NT*. Bis Mitte 2008 erwartet er aber „den großen Run in Sachen Betriebssystemwechsel".[1]

„Für den normalen Anwender ändert sich wenig", hat Ude seine ersten Kontakte mit dem neuen Softwarepaket zusammengefasst. Die Frankfurter Rundschau zitiert eine enge Mitarbeiterin des SPD-Politikers, die seit dem Start der inzwischen abgeschlossenen Pilotphase im Juli „den OB nicht über seinen Rechner fluchen hören" habe. Die Verwaltungsrätin selbst vermisst aufgrund eigener Erweiterungen die Tabellenkalkulation *Excel* aus *Microsoft Office*, mit der sie die Termine ihres Chefs verwaltet habe. Andererseits biete das neue System „nette zusätzliche Funktionen" wie gelbe Notizzettel. Insgesamt könne sie weder eine Verbesserung noch eine Verschlechterung feststellen. Oft in Anspruch genommen habe sie anfangs die extra eingerichtete, zu Bürozeiten ständig besetzte Hotline. Die Nummer zur Verhinderung von Fehlstarts in die leicht veränderte Computerwelt steht auf einem Aufkleber an ihrem Schreibtisch, der das Pinguin-Maskottchen „Mux" in einem Flieger zeigt. Nach wenigen Tagen habe sie sich aber mit den Neuerungen angefreundet gehabt.

„Obwohl einige Standardprogramme sehr ähnlich sind, kann es in den ersten Tagen Anlaufprobleme geben, weil die Menschen eine ganz bestimmte Erwartungshaltung gegenüber dem Arbeitsgerät Computer entwickelt haben", erklärt Jens Barth, Leiter des Teilprojekts Linux-Client. „Aber das wird sich schnell legen und sie werden merken, dass die Computer ihre Aufgaben genauso – oder in manchen Belangen sogar besser – erledigen und auch einfach zu bedienen sind." Der Basisclient, für dessen Programmieren die mittelständischen Firmen *Softcon* und *Gonicus* im April 2005 den Zuschlag nach einer europaweiten Ausschreibung erhalten hatten, setzt vor allem auf der Linux-Distribution *Debian GNU/Linux 3.1* auf und nutzt den Desktop *KDE 3.5* sowie *OpenOffice.org 2.0*. Als E-Mail-Programm kommt *Thunderbird*, zum Surfen *Firefox* zum Einsatz.

Mit der Software *Wollmux*, einem System für Textbausteine und Briefvorlagen, werden sich laut Barth zudem künftig Dokumente komfortabler erstellen und For-

1 Zur Linux-Migration der Stadt München siehe auch den Artikel von Wilhelm Hoegner (2006) im Open Source Jahrbuch 2006.

mulare einfacher ausfüllen lassen als bisher. Das Programm ist in *Java* geschrieben und somit plattformunabhängig ausgerichtet, wobei es im LiMux-Client konkret mit *OpenOffice.org* zusammenarbeitet. Es zieht die speziellen Benutzer- und Organisationsdaten aus einem zentralen Verzeichnis und erstellt Kopfbögen einem übergreifend vorgegebenen Corporate Design gemäß. Auch eigene Anpassungsarbeiten durch die Entwickler sind Barth zufolge leichter zu bewerkstelligen. Darüber hinaus gebe es natürlich noch das politische, vom Stadtrat ausgegebene Ziel, dank Linux „unabhängig zu sein von Monopolisten wie etwa Microsoft".

2 Applikationen wandern ins Web

Der Weg bis zur Umrüstung der ersten PCs auf freie Software war lang, mühsam und schwer umkämpft. „Wir haben den Aufwand unterschätzt, sind in manchen Punkten nicht so schnell vorangekommen und von manchen Unterstützern enttäuscht worden", erinnert sich Hoegner nach der anderthalb Jahre dauernden Entwicklung des Clients. Er habe manchmal Momente, in denen er sich frage, „was ich mir für Ärger hätte ersparen können". Detailprobleme habe es etwa bei der Steuerung der Infrastruktur, der Verteilung der neuen Applikationen und der automatischen, skriptgesteuerten Übernahme von Profilen für einzelne Nutzergruppen mit unterschiedlichen Zugriffsrechten auf die Rechner gegeben. Mit viel Aufwand sei nun aber ein tragfähiges Konzept vorhanden, um eine Migration in dieser Größenordnung zu bewältigen. Das Aufspielen des Clients und die Rechteverwaltung seien inzwischen „optimal steuerbar".

„Andere Sachen laufen von Anfang an überraschend gut", freut sich Hoegner. So seien etwa viel mehr Firmen bereit, Linux allgemein als Alternative auf dem Desktop zu sehen und auf Web-Applikationen zu setzen. München habe hier insbesondere bei Firmen mit Fachanwendungen, die als besonderes Hindernis bei jeder Migration hin auf Open Source gelten, eine „Eisbrecherfunktion" gehabt. Die Verwaltung selbst sei bei einer Zählung vor der Umstellung auf 120 eigene und 160 fremde Fachverfahren gekommen. Von den insgesamt 280 Anwendungen liefen Hoegner zufolge Anfang 2007 bereits rund die Hälfte unter Linux. Dazu würden Großrechnerverfahren genauso zählen wie E-Mail, Terminkalender, Bürosoftware, Texteditoren oder Geoinformationssysteme für die Vermessungstechnik. Besonders freut sich der Elektrotechniker, dass *SAP* inzwischen betriebssystemunabhängige Clients in der Software *ERP5* mit identischem Funktionsumfang wie bisher zur Verfügung stelle. Damit sei eine gleichberechtigte Integration von *OpenOffice.org* mit dem Warenwirtschaftssystem möglich.

Ziel ist es nun in München, neue Fachapplikationen möglichst nur noch im Web zu haben. Aber auch für alle alten Verfahren gebe es eine Umstellungsstrategie, versichert Hoegner. Notfalls müsse man dabei Terminallösungen als Brücken zwischen proprietären und offenen Welten einsetzen, obwohl diese verhältnismäßig teuer sei-

en, gewisse Netzbandbreiten benötigten und nur mit hohem Personalaufwand an den Start zu bekommen und zu pflegen seien. Einzelne Firmen wie *Autodesk* oder *Esri* würden sich zudem schlicht weigern, Lizenzen für den Terminalbetrieb ihrer Konstruktionssoftware zu erteilen. „Ich vermute sehr stark, dass Microsoft dabei seine Finger im Spiel hat", ärgert sich Hoegner. Die Umstellung der entsprechenden Softwarebestandteile von 20 Arbeitsplätzen im Designbereich werde daher wohl am längsten dauern. Ansonsten gebe es noch die Möglichkeiten, mit Emulatoren oder Virtualisierungssoftware wie *WINE* oder *VMware* zu arbeiten für Übergangslösungen, bis native Linuxanwendungen oder Java-orientierte Applikationen auf dem Client oder Browser-orientierte Serververfahren vorhanden seien.

3 Ballmer ante Portas

Trotz der Reibereien wegen der CAD-Software bezeichnet Hoegner das Verhältnis zu Microsoft inzwischen als entspannt bis kaum noch vorhanden. Der Softwaregigant selbst habe die Münchner Migrationspläne Anfang 2003 mit Symbolwert versehen, indem Geschäftsführer Steve Ballmer kurzfristig aus der Schweiz zu einem mehr oder weniger diktierten Gespräch mit dem Oberbürgermeister ins Rathaus geeilt sei. Zuvor habe es in der Deutschland-Filiale in Unterschleißheim direkt vor den Toren Münchens noch geheißen, „wegen eurer 10 000 Arbeitsplätze machen wir uns doch nicht krumm", rekapituliert Hoegner. „Ihr habt alles auf *NT* fokussiert, da könnt ihr eh nicht runter", habe es damals geheißen. Zuvor habe er bei der Vorstellung eines neuen Preismodells für Microsoft-Software, bei dem die Verwaltung sehr viel mehr als bisher hätte bezahlen müssen, noch halb scherzhaft gefragt, ob man damit rechne, dass die Stadtoberen die Alternative Linux näher beleuchten könnten. Erst als der Stadtrat eine wirtschaftliche Prüfung eben einer solchen weitgehenden Ablösung von Microsoft verlangte und eine gesamtwirtschaftliche Betrachtung des Beratungshauses *Unilog* mit den Stadtreferaten die Risiken eines Wechsels für überschaubar hielt und bei einer strategischen Betrachtung Linux empfahl, habe dies Ballmer auf den Plan gerufen und das Vorhaben ins Licht der Öffentlichkeit gerückt.

Als „irritierend" empfindet Hoegner noch immer die sture Haltung des für seine Hartnäckigkeit bekannten Microsoft-Geschäftsführers, nicht im Lizenzpreis nachgeben zu wollen. Stattdessen habe Ballmer kostenlose Personalleistungen etwa für die Umrüstung von Verzeichnisdiensten angeboten, die in München aber eh nicht vorhanden gewesen seien. Erst später sei noch die Offerte gekommen, die Textverarbeitung *Word* im größeren Umfang separat ohne Bündelung mit Microsofts kompletten Office-Paket zu lizenzieren. An dieser Stelle seien aber die Open-Source-Verfechter *IBM* und *SUSE* ins Spiel gekommen, die mit den Nachweis erbringen wollten, dass Linux auf einem Verwaltungsarbeitsplatz funktioniert.

Genau erinnert sich der EDV-Chef zudem an den „einmaligen Vorgang", dass Microsoft bei der entscheidenden Stadtratssitzung „in den Vorräumen anticham-

brierte“ und die Stadtväter in letzter Minute doch noch zu einer Migration hin zu *Windows XP* überzeugen wollte. Dies habe zumindest in der SPD aber die Ansicht verfestigt, dass eine strategische Lösung mit dem Versprechen von größerer Unabhängigkeit von einem einzelnen Hersteller und besseren Steuerungsmöglichkeiten der verwendeten Software und ihrer Updates der vorteilhaftere Weg sei. Zudem hätten Sicherheitsbedenken gegen die Microsoft-Software mit den Ausschlag gegeben. Die Redmonder haben Hoegner zufolge „keinen Nachweis zur Eignung ihres Betriebssystems für die sachgerechte Verarbeitung personenbezogener Daten erbracht, die bei uns aber Tagesgeschäft ist.“

Nach dem Grundsatzbeschluss zur Migration, dem sich allein die CSU aus Protest gegen das rot-grüne Münchner Regierungsbündnis widersetzte, sei der Kontakt zu Microsoft zunächst ziemlich zusammengebrochen, berichtet Hoegner weiter. Erst nach einem halben Jahr habe sich wieder einmal jemand aus Unterschleißheim gemeldet und gefragt, „ob wir überhaupt weiterkommen“. Microsoft habe sich dann sogar neben 33 Kleinunternehmen in der ersten Stufe am Ausschreibungsverfahren für den LiMux-Client beteiligt, ergänzt Projektmitarbeiter Schießl. In diesem Rahmen hätte der Konzern die Stadtverwaltung bei allen verbleibenden Problemen und bei der Komplettintegration der Migration unterstützen wollen – aber natürlich rund um Windows-Arbeitsplätze. Damit wäre München aber nicht gedient gewesen, begründet Schießl das Ausscheiden Microsofts vor der zweiten Runde. Der damalige Chef von Microsoft Deutschland habe später zugestimmt, im Bereich Terminal-Services und Fachverfahren stärker mit München zusammenarbeiten zu wollen, führt Hoegner weiter aus. Letztlich habe er aber nur Lizenzkosten mitgeteilt „und nichts hingestellt an Terminalservern“.

4 Stolperfalle Vergaberecht?

Gleichzeitig sei von Microsoft die Klage erhoben worden, in den Ausschreibungen für die Migration benachteiligt worden zu sein, kommt Schießl auf eine weitere Finte der unterlegenen Partei zu sprechen. Zu dieser Zeit, Ende 2004, versuchte Microsoft in Kooperation mit dem Bundesverband Digitale Wirtschaft, generell eine Debatte in Gang zu bringen, wonach die Einführung freier Software in der Verwaltung übers Vergaberecht stolpern könnte. So vertrat Dirk Heckmann (2004), Professor für Öffentliches Recht an der Universität Passau, die These, dass staatliche und kommunale Verwaltungen „nicht wissentlich Anbieter ausschließen dürfen“. Eine ausdrückliche Anforderung von Produkten wie Linux oder eine bedarfsorientierte Leistungsbeschreibung widerspreche „im Regelfall“ dem Gebot einer neutralen Ausschreibung. München warf er vor, „keine Bedarfsermittlung im Vorfeld“ durchgeführt zu haben. Open-Source-Verfechter hielten dagegen, dass die Diskussion nach hinten losgehe, da in der Praxis in 60 Prozent der Ausschreibungen proprietäre Lösungen verlangt würden.

Für München stellt Schießl ferner klar: „Wir hatten drauf geachtet, dass wir nicht Open Source in die Ausschreibung schreiben. Wir wollten bewusst nicht in eine Rechtsunsicherheit tappen." Bei einem Vortrag Heckmanns im bayerischen Innenministerium habe er den Professor auf seine Thesen hin angesprochen und die Auskunft erhalten, dass dieser „keine unserer Ausschreibungen gelesen hat". Heckmann habe eingeräumt, Angebote von Microsoft zur Finanzierung von Studien in diese Richtung erhalten zu haben, diese aber nie angenommen zu haben.

Als „Riesensauerei" bezeichnet Hoegner letztlich den Versuch einer Führungskraft in der Berliner Senatsverwaltung im Sommer 2006, die Linux-Migration als Fehlschlag zu bezeichnen. Der Berliner IT-Staatssekretär Ulrich Freise hatte damals im Abgeordnetenhaus die angebliche Ansage eines hochrangigen Beamten der Stadt München zum Besten gegeben, dass „wir leider an der Wirklichkeit gescheitert sind". Keineswegs würden demnach alle Anbieter ihre Fachverfahren neu stricken. Alles andere sei Wunschdenken und vergleichbar damit, dass „Coca-Cola seine Rezeptur" freigibt. „Wir haben versucht, zu recherchieren, wo das herkommt", erklärt Hoegner. Zu einem klaren Ergebnis sei man bis heute aber nicht gekommen. Auf eine Beschwerde unter Genossen an den Regierenden Bürgermeister von Berlin, Klaus Wowereit, sei nur ein „nichtssagendes Schreiben zurückgekommen". Zuvor habe es eine Anfrage aus der SPD-Fraktion im Berliner Abgeordnetenhaus gegeben in Richtung Erfahrungsaustausch. Daraufhin habe man die aktuellsten Vorträge und Unterlagen zur Verfügung gestellt. Aus einem angekündigten Besuch der Berliner Kollegen in München sei aber nichts geworden.

Die Linux-Revoluzzer aus der bayerischen Hauptstadt bekamen zudem mit, dass Microsoft den ehemaligen Berliner Wirtschaftssenator Wolfgang Branoner (CDU) verpflichtete, um nicht nur im Senat der Bundeshauptstadt, sondern allgemein im öffentlichen Bereich auf Länderebene Lobbying für Windows und Redmond zu betreiben sowie Verkaufsgespräche zu führen. Es sei sonst nicht üblich, jemand aus dem öffentlichen Dienst für einen solchen Job anzuheuern, schüttelt Hoegner den Kopf über diesen weiteren Schachzug. In anderen Städten sei Microsoft dann auch sehr wohl mit dem Preis heruntergegangen, weiß der EDV-Chef von Rechenzentrumskollegen. „Die sind uns sehr dankbar, dass wir das auf die Spitze getrieben haben", sagt Hoegner mit einem lachenden und einem weinenden Auge, da damit der Druck zum Wechseln auf Linux in anderen Kommunen schwächer werde. Insgesamt habe Microsoft begonnen, das Open-Source-Lager ernster zu nehmen und von der „Überheblichkeit" aufgrund des hohen Marktanteils etwas abzugehen. „Die haben eine ganze Reihe von Aktivitäten entwickelt, *XP* beispielsweise gründlich in Sachen Sicherheit durchleuchtet und deutlich nachgebessert", konstatiert Hoegner. „Das reicht vielleicht nicht aus, aber sie haben einen deutlichen Schwerpunkt auf eine ordentliche Benutzerverwaltung und die Abgrenzung von Funktionen gelegt."

2007 ist für Schießl dennoch das „Jahr der Migration", da es nun um *Vista* oder eine Alternative gehe. „Es wird Bewegung geben", ist sich der Linux-Experte sicher.

Viele würden sich jetzt überlegen, „einen weiteren Schritt in Richtung Unabhängigkeit zu gehen“ und Linux nach dem Serverbereich auch auf dem Desktop in Erwägung zu ziehen. Bis zur Mitte des Jahres werde man daher genau darüber nachdenken, inwieweit es sinnvoll wäre, den LiMux-Client und *Wollmux* unter der *GNU General Public License (GPL)* zu veröffentlichen und anderen Gemeinden damit den Schritt zur Loslösung von Windows zu erleichtern. Problematisch sei dabei nur, dass beide Programme auf München maßgeschneidert und nicht gänzlich auf die Belange anderer Kommunen zu übertragen seien.

5 IT-Strategie vorangebracht

Hoegner zieht derweil für das Münchner Großprojekt insgesamt ein positives Resümee: „Wir tun mehr für die städtische IT-Strategie, als nur ein Betriebssystem auszutauschen. Es kommt zu einer viel stärkeren Standardisierung und Vereinheitlichung.“ Dinge, die sich über die Jahre hinweg auseinanderentwickelt hätten, seien nun wieder zusammengefügt worden, fügt Schießl hinzu. Möglichst viel der Informationsverarbeitung sei ins Web verlagert und plattformunabhängig gestaltet worden, „damit wir in zehn Jahren bei einer neuen Migration nicht vor denselben Problemen stehen.“ Es gebe nun eine „verteilte Infrastruktur, die viel stärker zukunftsorientiert ist.“

„Unser Ziel war immer, dass wir selber im Griff haben, wann wir Betriebssysteme updaten wollen“, wirft auch Hoegner einen Blick nach vorn. Dabei sei man bei der Anbindung an die Hardware zwar nicht ganz unabhängig und müsse natürlich auch am Linux-Kernel immer wieder Sachen ergänzen. Dies sei aber nicht damit verbunden, „auch gleich die Version von *KDE* ändern zu müssen.“ So könne man das System länger stabil halten, die Zyklen ein wenig verlangsamen und zumindest den Zeitpunkt für ein Update selbst bestimmen. Bei Microsoft sei man dagegen gezwungen, „immer gleich alles auszutauschen.“ Eine sofortige Ersparnis bei der Hardware sieht Schießl dagegen nicht, da besonders die graphische Benutzeroberfläche auf die Leistung gehe. „Unsere jetzigen Installationen brauchen zwar weniger Ressourcen als *Vista*, aber die gleichen wie *XP*“, führt er aus. Unschlagbar sei Linux aber bei Webservices und Kommunikationsdiensten. Hier könne man „ganz schlanke Distributionen erstellen, die auch für veraltete Maschinen zu gebrauchen sind.“

Bei den Kosten sieht Hoegner eine Chance, dass sie unterhalb der insgesamt veranschlagten 35 Millionen Euro bleiben. Der Löwenanteil von 38 Prozent davon ist für Schulungen der Mitarbeiter vorgesehen, die sich aber als weniger aufwendig gestalten als befürchtet. „Was habt ihr für ein Gschiss gemacht“, zitiert der EDV-Chef Angestellte, denen ein ganzer Tag Umschulung deutlich zu lang war. Bürgermeisterin Strobl zeigte sich bereits im Sommer 2006 nach ersten Klicks in der neuen Desktop-Umgebung von deren Benutzerfreundlichkeit überrascht: „Das sieht jetzt wirklich nicht so anders aus, als dass ich mich umschulen lassen müsste“, freute sich die bekennende Nicht-Computerexpertin.

Den IT-Fachmann Hoegner plagen derweil höchstens noch die von Microsoft immer wieder geschürten Sorgen um gewerbliche Schutzrechte rund um Linux. Die wichtige Ausschreibung des Basisclients hatten die Münchner bereits im Hochsommer 2004 um wenige Wochen verschoben. Damals hatte es in der Stadtverwaltung Bedenken zu den potenziellen Auswirkungen der geplanten, letztlich vom EU-Parlament aber abgelehnten EU-Richtlinie über die Patentierbarkeit von Software gegeben. Ein Rechtsgutachten habe der Verwaltung aber die formale Rückendeckung gegeben, das Projekt weiterzuführen, blickt Hoegner zurück. Demnach seien die Risiken durch Patentverletzungen für proprietäre Software und Open-Source-Programme gleich hoch bewertet worden. Zumindest habe der kurzzeitige Stopp aber bewirkt, dass die Thematik der Softwarepatente weit grenze und nicht nur für die von der CSU belächelten „Feierabendprogrammierer" freier Software heikel sei.

In diesem Zusammenhang beäugt Hoegner die „Nichtangriffspakte in Sachen Patente" zwischen Microsoft und *Novell* beziehungsweise *Sun Microsystems* skeptisch. Bei Ersterem etwa werde nur der Novell-SUSE-Desktop von Patentansprüchen freigestellt, „aber nicht das, was an die Open-Source-Gemeinde abgegeben wird". So werde der Versuch unternommen, die Entwickler freier Software „methodisch immer weiter zu isolieren von ihren Förderern im Firmenumfeld." Derlei Abkommen könnten letztlich dazu dienen, zunächst in den USA den Open-Source-Bereich einzugrenzen „und dann mal zu sagen, da und da gibt es Verstöße gegen unsere Patente".

Dementsprechend behauptete Microsoft-Frontmann Ballmer im November 2006, dass Linux „unser patentiertes geistiges Eigentum nutzt". Man könnte gewissermaßen sagen, „dass jeder, der Linux in seinem Daten-Center betreibt, ein nicht genau beziffertes Minus auf seinem Konto hat", weitete Ballmer seinen Vorwurf der Patentverletzung auch auf alle Anwender des frei verfügbaren Betriebssystems aus. Angesichts der Funktionsweise von Open Source sei bislang niemand imstande gewesen, dafür eine Freistellung von Patentansprüchen anzubieten. Rechtsexperten aus der Open-Source–Gemeinde wie die Betreiberin des Blogs *Groklaw*, Pamela Jones, bezeichneten den neuen Angriff Ballmers auf Linux jedoch als „FUD" (*Fear, Uncertainty and Doubt*), als Streuen von Furcht, Unsicherheit und Zweifel. Der Microsoft-Chef möge doch bitte klagen, falls er sich wirklich sicher sei, Rechtsansprüche geltend machen zu können.

6 Kostendrücker Linux

Öffentliche Verwaltungen wie München gelten insgesamt als Treiber für den Einsatz von Open Source. „Der öffentliche Sektor setzt Open-Source-Software schon jetzt in überdurchschnittlichem Maße ein", weiß Jochen Günther vom Fraunhofer-Institut Arbeitswirtschaft und Organisation (IAO). Mit weiter steigender Verbreitung sei zu rechnen. Der Forscher hat Anfang 2007 eine Studie veröffentlicht (Günther 2006), die *IBM*, *Novell* und die *Wirtschaftsförderung Region Stuttgart GmbH (WRS)* in

Auftrag gegeben hatten. Für die Erhebung schrieben Günther und seine Mitarbeiter 2000 öffentliche Einrichtungen von lokalen Behörden an, in denen die Durchdringung mit IT besonders hoch ist, bis hin zu Bundesministerien. 115 davon antworteten auf den Fragebogen. Zugleich äußerten sich 94 angesprochene Geschäftsführer oder Inhaber von IT-Unternehmen zu dem Thema.

Im Bereich der Lizenz- und Betriebskosten führt Open Source laut den Ergebnissen der Untersuchung zu teilweise erheblichen Einsparungen. 47 Prozent der Teilnehmer gehen von Kostensenkungen von mehr als 50 Prozent aus. Weitere 20 Prozent würden mit Ersparnissen von bis zu 25 Prozent rechnen. Für fast 60 Prozent der Teilnehmer gehörte die Umstellung auf freie Software zu einer mittel- oder langfristigen IT–Gesamtstrategie mit dem Ziel, systematisch betriebskritische Teile der IT-Landschaft auf Open Source zu migrieren. Der wichtigste Einsatzzweck sei für die befragten öffentlichen Einrichtungen die Ergänzung der IT durch quelloffene Anwendungen vor allem in den Bereichen Office-Anwendungen, Telefonie-Infrastruktur und Desktop–Systeme. Die meisten neu eingeführten Open-Source-Lösungen würden dabei auf den Bereich der Fachverfahren entfallen, die verstärkt mit Hilfe Open-Source-basierter Middleware realisiert werden. Das Ende des Lebenszyklus von Fachanwendungen werde dabei teilweise gezielt genutzt, um eine Plattformunabhängigkeit von Applikationen durch deren webbasierte Umsetzung zu erreichen. In diesen Zyklus würden zudem Beschaffungsentscheidungen für neue Benutzer- oder Serverplattformen fallen, die nicht mehr an die Systemanforderungen der proprietären Alt-Systeme gebunden seien.

Aufträge der öffentlichen Verwaltungen bei der Migration zu Linux gehen der Studie zufolge bevorzugt an kleinere und mittlere regionale Dienstleister. Open Source führe dadurch auch verstärkt zu regionaler Wertschöpfung und zu einem erhöhten Innovationspotenzial der IT-Unternehmen. Generell böte sich durch die Förderung freier Software die Chance, dass aus öffentlichen Mitteln wieder „öffentliche Güter" in Form von Open-Source-Programmen erwachsen würden. Diese stünden dann sowohl öffentlichen Einrichtungen als auch IT-Unternehmen zur wirtschaftlichen Verwertung frei zur Verfügung. Mitte 2004 waren Pasche und von Engelhardt (2004) bereits zu dem Ergebnis gekommen, dass die Entwicklung von Open-Source-Software „direkt und indirekt zu einer Erhöhung der Wettbewerbsintensität im kommerziellen Sektor" beitrage. Die Autoren sahen in ihrer Forschung auch eine direkte Erwiderung auf eine Studie der Universität Münster aus 2003, die Microsoft als Jobmotor darstellte (Kooths et al. 2003*a*). In einer weiteren umstrittenen Analyse kamen die Münsteraner auch zu dem Schluss, dass freie Software aus volkswirtschaftlicher Sicht keine geeignete Alternative zum kommerziellen Softwaremarkt biete (Kooths et al. 2003*b*).

7 Das Open-Source-Ministerium

Als Musterbehörde im Sinne der Studie des IAO kann das Auswärtige Amt bezeichnet werden.[2] „Wir haben seit Januar 2002 die Strategie, Open Source überall einzusetzen, wo dies sinnvoll ist", erklärt der für die IT-Ausrichtung des Außenministeriums zuständige Leiter Rolf Schuster. Inzwischen laufe nur noch auf den stationären Desktops Windows, während die Devise laute, „praktisch nur noch freie Software einzusetzen". Ziel sei es, Lizenzkosten zu sparen und die von Microsoft diktierten Update-Zyklen nicht mehr mitzumachen. Als „legitim" bezeichnet es Schuster zwar, dass die Redmonder ihre Geschäftsinteressen verfolgen und weiter Lobbying bei der Verwaltung betreibe. „Aber wir müssen im Interesse des Steuerzahlers handeln", hält der IT–Experte dem entgegen. Bei *Vista* etwa kenne er keine maßgebliche Funktion, „die wir benötigen." Und die umfassenden Möglichkeiten des neuen *Microsoft Office* nutze eh keiner.

Die Erfolgsgeschichte von Linux im Auswärtigen Amt startete Schuster zufolge mit dem Ende 2001 begonnenen Anschluss von inzwischen 230 Auslandsvertretungen an ein hochsicheres diplomatisches Intranet. „Das war als 100-Millionen-Projekt angesetzt, aber bekommen haben wir nur 17 Millionen Euro", erinnert sich der IT-Stratege. Also musste an allen Ecken und Enden gespart werden. Die gewählte Vernetzungslösung basiert so ausschließlich auf Open-Source-Techniken. Die Grundlage bilden Linux-Server, sowie so genannte *SINA-Boxen* der Firma *Secunet*. Diese ebenfalls gemeinsam mit dem BSI entwickelte *Sichere IntraNetwork Architektur* sorgt für die höchsten Geheimhaltungsstufen gemäß zertifizierte Verschlüsselung des gesamten Datenverkehrs im Intranet des Auswärtigen Amtes. Zusätzlich ist in jeder Botschaft ein Linux-Router installiert, der die je nach örtlichen Gegebenheiten verfügbaren Fernverbindungen ausfallsicher bündelt. Dabei handelt es sich um Breitband-Anschlüsse, welche die Botschaften direkt bei einem örtlichen Provider anmieten, um hochverfügbare IP-Anbindungen oder Satellitenverbindungen. Mit diesen sollen die Auslandsvertretungen auch dann zu erreichen sein, wenn vor Ort die Leitungen gekappt werden.

Darüber hinaus hält Linux auch verstärkt auf mobilen Rechnern der Diplomaten Einzug. „300 bis 400 entsprechende Laptops sind bereits im Umlauf", freut sich Schuster. Als Basis komme die Debian-Distribution zum Einsatz, wobei die SINA–adäquate Kryptographie dieses Mal über ein Verschlüsselungsgerät mit einem USB–Smartcard-Token erfolge. An Verbindungstechniken würden GPRS, UMTS, WLAN und LAN-Anschlüsse unterstützt, bei den auf Ministerreisen verwendeten Geräten zudem tragbare Satellitenkommunikation. Dazu komme *OpenOffice.org* für die Textverarbeitung. „In einer virtuellen Maschine haben wir für den Notfall Windows noch mit drin, aber nur mit gekappten Netzverbindungen", erläutert Schuster das Konzept.

2 Siehe zur Open-Source-Strategie des Auswärtigen Amtes auch den Artikel von Torsten Werner auf Seite 239 in diesem Buch.

Dies sei nötig, da bei manchen 200 oder 300 Seiten umfassenden Regierungsdokumenten im DOC-Format in *OpenOffice.org* nicht alle Seitenumbrüche korrekt abgebildet würden, was im diplomatischen Alltagsgeschäft zu Problemen führen könnte. Schulungen für den Umgang mit den neuen Applikationen hätten sich nicht als erforderlich erwiesen. Eine zwanzigminütige Einweisung reiche in der Regel völlig aus.

Besonders froh ist Schuster, dass das Auswärtige Amt seit der Umpositionierung „das mit Abstand günstigste Ministerium im Bund bei den IT-Kosten ist, obwohl wir unter erschwerten Bedingungen arbeiten." 80 Prozent der betreuten Rechner stünden im Ausland, wo schon einmal der ein oder andere Sandsturm über die Botschaften hinwegfege und die PCs oft bereits nach drei Jahren nicht mehr zu gebrauchen seien. Trotzdem hätten die IT-Ausgaben 2005 pro Kopf im Jahr bei 1 190 Euro gelegen. Andere Ministerien hätten dafür mindestens das Doppelte und teilweise bis 5 000 Euro, der Bundesrechnungshof immer noch 2 700 Euro ausgegeben. Die enormen Unterschiede, ist sich Schuster sicher, „haben mit Open Source zu tun". Die anderen Ressorts hätten eben höhere Budgets für Informationstechnologie zur Verfügung, sodass sie bei Linux noch „mehr Zurückhaltung" an den Tag legen würden. Wichtig sei es daher, dass zumindest die Einhaltung offener Standards für die gesamte Bundesverwaltung verpflichtend werde.

Die Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) bearbeitet den Bereich Open Source und IT-Standards ressortübergreifend.[3] Dabei muss sie angesichts der Eigenverantwortung der Verwaltungsebenen nicht wenig Überzeugungsarbeit leisten. „Wir stellen Informationen über Open Source zur Verfügung und weisen Alternativen zu den gängigen Betriebssystemen auf", sagt ein Sprecher des Bundesinnenministeriums, zu dem die KBSt gehört. Gerade wenn eine Migration ins Haus stehe, würden die entsprechenden Verwaltungen dazu aufgefordert, auch über das „wohin" nachzudenken. Weitere Hilfestellungen seien ein Migrationsleitfaden sowie der Betrieb eines Open–Source-Kompetenzzentrums. Darin werden neben einer Darstellung ökonomischer und rechtlicher Aspekte frei verfügbarer Software auch inzwischen über 100 Projekte aus der Bundesverwaltung aufgelistet, in denen Open Source zum Tragen kommt. Ansonsten gelte aber das „Neutralitätsgebot", das gerade bei Ausschreibungen zu beachten sei.[4]

Generell nutzen dem Sprecher zufolge inzwischen „viele Ministerien und ihre nachgeordneten Behörden Open Source in irgendeiner Form". Das Bundeswirtschaftsministerium etwa hat *Firefox* als Browser auserkoren, während das BSI auf allen Arbeitsplätzen auch *OpenOffice.org* installiert hat. Das Ministerium, das am meisten freie Software im Einsatz habe und so das beste Praxisbeispiel abgebe, dass Linux im

3 Siehe zur Arbeit der KBSt auch den Artikel von Joachim Sturm (2005) aus dem Open Source Jahrbuch 2005.

4 Der Migrationsleitfaden der KBSt in der Version 2.1 findet sich unter http://www.kbst.bund.de/Content/Software/Migration/migration__node.html. Das Kompetenzzentrum betreibt ein Web-Portal unter http://www.kbst.bund.de/Content/Software/OSS__Kompetenzzentrum/oss.html.

Behördenumfeld gut funktioniere, ist laut dem Sprecher aber das Auswärtige Amt. Als größte Vorteile freier Software gelten bei der KBSt die Stabilität der Applikationen, die große Entwicklergemeinde, der Wegfall von Lizenzkosten, die Möglichkeit, Sicherheitslücken notfalls selbst zu schließen, und der langfristige Zugriff auf den Quellcode. Microsoft betreibe dagegen „ein sehr aktives Marketing und in konkreten Fällen durchaus offensiv Lobby-Arbeit."

8 Showdown in der Bundeshauptstadt

Kräftig kämpfen müssen die Microsoft-Lobbyisten, die inzwischen jedes bei Windows bleibende Landratsamt einer Pressemitteilung für würdig erachten, nach wie vor in Berlin. In der Bundeshauptstadt hatte der Hauptausschuss des Landesparlamentes im Dezember 2005 vom Senat einstimmig die Vorlage eines Fahrplans zur Migration auf Linux verlangt. Es sollte zunächst um die Umstellung der Server und in einem späteren Schritt auch um die Umrüstung der rund 58 000 in der Hauptverwaltung benutzten Arbeitsplatzcomputer auf alternative Betriebssysteme sowie Desktop- und Anwendungssoftware aus dem Open-Source-Bereich gehen. Weiter pochen die Abgeordneten darauf, dass bei der Beschaffung neuer Hardware auf die Eignung für die Verwendung freier Software „ohne Einschränkungen" geachtet wird.

Insgesamt soll die Berliner IT-Landschaft dem Bestreben der Parlamentarier nach offener werden, wovon sie sich mehr Wettbewerb, weniger Abhängigkeiten und letztlich Kostenersparnisse erhoffen. „Es muss sichergestellt sein, dass bei der Beschaffung von Software offene Standards (Schnittstellen, Dokumentenformate) als Beschaffungskriterium gegeben sind", hielten die Abgeordneten fest. Bei allen Softwarekäufen und bei der Erstellung eigener IT-Lösungen sei „strikt auf Schnittstellen- und Dokumentenformatoffenheit" zu achten. „Tatsächliche Standards, wie beispielsweise *Open Document Format (ODT)* oder *Extensible Markup Language (XML)*" müssten zudem im Gegensatz zu „herstellerabhängigen Quasi-Standards" den Beschaffungs- und IT-Lösungsansätzen zu Grunde gelegt werden.

Zuvor hatte sich Microsoft mit einem bereits im Vorfeld heftig umstrittenen Lobby–Abend direkt im Berliner Abgeordnetenhaus Mitte August 2005 in die Nesseln gesetzt. Unter die etwa 50 der 141 Berliner Parlamentarier, die sich für das von einem Büfett gekrönten Informations- und Gesprächsangebot angemeldet hatten, mischten sich damals auch drei als Pinguine verkleidete Linux-Aktivisten. Die zum Dessert gereichte Mousse au Chocolat krönten zudem kleine Fähnchen, die mit den Aufdrucken „Alt+F4" zum Schließen eines Fensters unter Windows und „Monokultur" versehen waren. Parlamentspräsident Walter Momper stand in seiner Begrüßungsrede aber zu seiner Entscheidung, Microsoft den Festsaal des Hauses für die Konzernveranstaltung mit den Abgeordneten zu vermieten.

Einer der Auslöser für die hitzigen Aktivitäten des Softwaregiganten: Kurz zuvor hatte die Senatsverwaltung für Inneres eine Umstellung in der Hauptverwaltung von

Microsoft-Software auf Open Source grundsätzlich für machbar erklärt. Linux sei mit geringfügig niedrigeren Betriebskosten als Windows zu betreiben, so die Ansage. Die Lauffähigkeit und die Kosten für die Umstellung der Fachverfahren seien aber als „kritischer Entscheidungsfaktor" anzusehen. Insbesondere der Ansatz einer „reinen Open-Source-Software-Landschaft" bis zum Endgerät erschien der Behörde zweifelhaft.

Die Senatsverwaltung erklärte den Beschluss des Abgeordnetenhauses ein Jahr später trotzdem dann mit deutlicher Verspätung zu der geforderten Zeitplanerstellung für „nicht umsetzbar" in all seinen Konsequenzen. Die angemahnte Migration hin auf freie Software erscheine „als nicht marktkonforme, technologisch und insbesondere wirtschaftlich nicht vertretbare Maßnahme", konstatierte ein entsprechender Bericht zur Position des Parlaments im Juni 2006. Die Abgeordneten konnten darin nachlesen, dass die „vollständige Umstellung" der Server- und Arbeitsplatzsoftware auf Linux eine „völlige Abkehr" vom Berliner „Open-Systems-Konzept" darstelle. Dieser „vom Senat verfolgten und dem Parlament bereits in diversen Berichten dargelegten Strategie" gemäß sei die Hauptaufgabe die Einrichtung „offener IT-Architekturen". Erreicht werden soll damit „die für einen wirtschaftlichen IT-Einsatz erforderliche Herstellerunabhängigkeit und Offenheit gegenüber unterschiedlichen Lösungen".

Auch der Abschlussbericht des auf Bezirksebene zuvor durchgeführten „Open4Future"-Projekts brachte den Senat nicht von seiner Meinung ab. Die Analyse hatte zum Ziel, das in einer früheren Machbarkeitsstudie empfohlene Szenario einer gemischten Open-Source-Umgebung „unter Einsatz der Terminalserver-Technologie" in Tempelhof-Schöneberg auszuloten. Dabei konnte die technische Umsetzbarkeit einer Lösung mit freier Software, Microsoft- und Novell-Produkten nachgewiesen werden. Auch die vorhandene „veraltete" Hardwareausstattung war weiter nutzbar. Das Pilotprojekt demonstrierte überdies an *IPV*, *ProFiskal* und *ProInfo*, dass die Migration von Fachverfahren keine unüberwindbare Hürde darstellt. Eine Orientierung an offenen Standards und Normen sei daher künftig die wichtigste Frage bei der Realisierung von IT-Projekten, lautet das Resümee der Studie. „Wenn sich auch ein bisher anders aufgestellter Marktführer so orientiert, bleibt uns nur, von dann aus allen Anbietern – bei gleichen Leistungsdaten – den wirtschaftlichsten auszuwählen".

9 Die Berliner Senatsverwaltung und ihre Sprechzettel

Der Senat verwies dagegen darauf, dass in der gesamten Berliner Verwaltung etwa 390 Fachverfahren eingesetzt würden und nicht alle umgestellt werden könnten. Zudem seien „Aufwandsschätzungen und Wirtschaftlichkeitsberechnungen" im Rahmen des „Open4Future"-Projekts nicht berücksichtigt worden. Die Studie könne so nicht als Voraussetzung für landesweite Planungen gelten. In der SPD-Fraktion gab sich die Basis jedoch nicht zufrieden mit der Stellungnahme des Senats. „Die Schlange windet und windet sich", war dort zu vernehmen. Nach „einem Vierteljahr Arbeit hoch

bezahlter Beamter und zusätzlicher Experten" sei ein „armseliges Papier" herausgekommen. So werde etwa die Tatsache ignoriert, dass irgendeine Migration weg von *Windows NT4* stattfinden müsse. Gleichzeitig falle kein Wort über die Kosten des Hardware-Ersatzes von mindestens einem Drittel aller Berliner Verwaltungscomputer im Fall einer Migration nach *Windows XP* oder *Vista* und des dabei ebenfalls erforderlichen Anwendungstransfers. Es gehe im Beschluss der Abgeordneten auch nicht um eine vom Senat unterstellte „Hau-Ruck-Umstellung", sondern um mittelfristige Vorgaben für unbedingt notwendige Neubeschaffungen und Updates.

In diesem Sinne stimmte der Ausschuss für Verwaltungsreform und Kommunikations- und Informationstechnik des Berliner Abgeordnetenhauses im Juni 2006 mit den Stimmen der regierenden rot-roten Koalition gegen einen Antrag der Grünen, der eine zügige Umstellung der Server-Betriebssysteme auf Open Source erreichen wollte. „Natürlich ist der Auflagenbeschluss in der Welt", betonte Kirsten Flesch, die für die SPD-Fraktion im Verwaltungsausschuss sitzt, während der heftigen Debatte. Er werde sicher auch im nächsten Haushaltsgesetz stehen und den Senat dadurch durch die Finanzzuteilung binden. Es gehe aber um eine strategische Zukunftsausrichtung, die sukzessive umgesetzt werden solle: „Wo es wirtschaftlich unter Vollkostengesichtspunkten möglich ist, wollen wir über offene Standards kommunizieren." Bei einem geschlossenen System und nicht zusammenpassenden Schnittstellen dürfe es nicht bleiben. Eine Mehrheit erhielt so der Vorschlag von SPD und Linkspartei, zunächst im Rahmen einer Anhörung das weitere Vorgehen rund um eine Öffnung der Computerplattformen in der Hauptstadt durchzuführen. Einen Termin dafür gab es bis Ende Februar 2007 aber noch nicht.

Der Berliner IT-Staatssekretär Freise versuchte im Rahmen der heftigen Diskussion, die zaudernde Haltung des Senats zu untermauern. Würde das Begehr des Hauptausschusses „im O-Ton" umgesetzt, würde Berlin seiner Ansicht nach „gegen das Haushalts- und Vergaberecht" verstoßen, knüpfte er an die von Microsoft schon gegenüber München vorangetriebene Argumentationsweise an. Gleichzeitig machte Freise die den Genossen in der bayerischen Landeshauptstadt so übel aufgestoßene falsche Bemerkung mit dem angeblichen Scheitern der Linux-Migration bei den Pionieren im Süden.

Als „traurig" empfindet es Gunnar Stöcker, ehemaliger Leiter der Berliner Geschäftsstelle zur Koordinierung und Beratung bezirklicher IT-Verfahren (KoBIT), nach wie vor angesichts dieses Vorfalls, „dass Staatssekretäre von Sprechzetteln leben". Das entsprechende Memo eines eigentlich für Ausschreibungsrecht zuständigen und damit nicht so stark mit IT-Lösungen erfahrenen Referatsleiters hat der Vorruheständler genauso in der Schublade wie ein „vertraulich" gekennzeichnetes „internes Arbeitspapier" von Microsoft zu der von ihm betreuten Machbarkeitsstudie für den Einsatz freier Software in der Bezirksverwaltung. Das 22-seitige Werk lässt naturgemäß kein gutes Haar an den Empfehlungen der KoBIT, da diese verschiedene Gebiete „nur unvollständig" und andere gar nicht betrachtet habe. Natürlich dürfe man den Wett-

bewerb nicht unzulässig einschränken, lautet Stöckers Devise. Aber man müsse offene Standards und Normen etwa den ISO-Festsetzungen gemäß vorgeben. Nur wer diese einhalte, könne mitspielen. Unverzeihlich sei es dagegen, wenn die Senatskanzlei in einer Tischvorlage vorschlage, den ISO-Standard für Zeichensätze für Berlin nicht mehr für verbindlich zu erklären, weil Microsofts *Internet Explorer* dazu nicht kompatibel sei. „Wir müssen uns lösen von Anbietern und Herstellern, die uns beherrscht haben", hält der IT-Experte derlei Ansinnen entgegen. Mehr Offenheit sei gefragt, etwa auch, wenn *Novell* sich im Netzwerkbereich eine vorherrschende Präsenz zu schaffen suche. Krude sei daher das Pochen auf das Vergaberecht: „Ich musste für jeden Bleistift drei Angebote einholen, aber dann verschwenden wir Geld für Microsoft, obwohl wir die Computer nur als bessere Schreibmaschine gebrauchen."

Stöcker widerspricht zudem der Behauptung Freises, wonach der Senat den Auflagenbeschluss des Parlaments schon „in annähernd allen Punkten" umsetze und zur Strategie der Herstellerunabhängigkeit stehe. Er verwies darauf, dass der KoBIT bei der Umsetzung der Ergebnisse des Projekts *Open4Future*[5] Steine in den Weg gelegt würden. In einem Parallelbetrieb mit der vorhandenen Technik wollte die Einrichtung ursprünglich noch 2006 genaue Wirtschaftlichkeitszahlen vorlegen. Weder Verantwortliche in dem ausgesuchten Bezirk noch in der Finanzverwaltung hätten die Tester aber ans Werk gelassen, da Anwendungen teilweise nur für einen Windows-Client freigegeben seien, beklagt Stöcker. Er spricht von einem „Tippel-Tappel-Weg" in Richtung offener Standards, da sich in Berlin neben dem Senat und dem Abgeordnetenhaus anders als etwa in München auch noch zwölf Bezirke zusammenraufen müssten.

Als letzten Handstreich vor seinem Ausscheiden aus der KoBIT schaffte es Stöcker Anfang 2007 noch, zumindest ein Pilotprojekt zur Nutzbarkeit von *OpenOffice.org* im Vergleich zu Microsofts Bürosoftware im Bezirksamt Tempelhof-Schöneberg an den Start zu bekommen. Dabei sollen innerhalb von vier bis sechs Wochen Erfahrungen der jeweiligen Nutzer dokumentiert werden. Leitlinien des Projekts sind die Kostenminimierung, die Herstellerunabhängigkeit und die Plattformneutralität beim Einsatz neuer Office-Lösungen. Zudem soll auf die strikte Einhaltung offener Standards geachtet werden. Bedeutsam ist der Test laut der Projektskizze, weil der überwiegende Anteil der Arbeitsplatzrechner in der Berliner Verwaltung noch *Windows NT4.0* als Betriebssystem und *Microsoft Office 97* für die Bürokommunikation nutze und damit Aktualisierungen aufgrund mannigfaltiger technischer und prozessualer Anforderungen unvermeidbar seien.

Stöcker hofft, dass das Votum der vorgesehenen rund 14 Berliner Tester sich *OpenOffice.org* zuneigt. Freie Software ist für ihn „nicht nur ein IT-Phänomen, sondern hat einen hohen gesellschaftlichen Anspruch. Wir kommen weiter, wenn wir Dinge offenlegen, jeder zur Verbesserung beitragen kann und die Entwicklungen für

5 Der Projektbericht findet sich unter http://www.parlament-berlin.de/ados/Haupt/vorgang/h15-3130.C.Anlage-v.pdf.

alle zur Verfügung gestellt werden." Auch Thomas Birk von der Fraktion der Grünen im Berliner Abgeordnetenhaus erwartet, dass der Senat sich nicht nur pro forma vom Umzug des Linuxtags – als eines der auch politisch maßgeblichen Open-Source–Stelldicheins – nach Berlin begeistert gezeigt habe. Vielmehr müsse nun eine echte „IT-Standortstrategie" mit einem Bekenntnis zu freier Software folgen.

Literatur

Günther, J. (2006), Open Source Software – Strukturwandel oder Strohfeuer?, Trendstudie, Fraunhofer Institut Arbeitswirtschaft und Organisation. Herausgegeben durch Dieter Spath. http://www.iao.fraunhofer.de/d/oss_studie.pdf [25. Feb 2007].

Heckmann, D. (2004), Open Source – Closed Shop? Vergaberechtliche Anforderungen an behördliche IT-Ausschreibungen zugunsten von Open Source Software, Wissenschaftliches Rechtsgutachten, Universität Passau. Im Auftrag der Microsoft GmbH, http://www.innrego.de/osvergabegutachten.pdf [25. Feb 2007].

Hoegner, W. (2006), Das Projekt Limux – Freie Software für die Münchner Verwaltungsclients, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 59–72. http://www.opensourcejahrbuch.de [25. Feb 2007].

Kooths, S., Langenfurth, M. und Kalwey, N. (2003*a*), Die Bedeutung der Microsoft Deutschland GmbH für den deutschen IT-Sektor (Economic Impact Study), MICE Economic Research Study Volume 3, Westfälische Wilhelms-Universität Münster, Muenster Institute for Computational Economics. http://mice.uni-muenster.de/mers/mers3-EconomicImpact_de.pdf [25. Feb 2007].

Kooths, S., Langenfurth, M. und Kalwey, N. (2003*b*), Open Source-Software: Eine volkswirtschaftliche Bewertung, MICE Economic Research Study Volume 4, Westfälische Wilhelms-Universität Münster, Muenster Institute for Computational Economics. http://mice.uni-muenster.de/mers/mers4-OpenSource_de.pdf [25. Feb 2007].

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 [25. Feb 2007].

Sturm, J. (2005), Die Open-Source-Strategie der öffentlichen Verwaltung, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 17–23. http://www.opensourcejahrbuch.de [25. Feb 2007].

World Domination*: Die Erfolgsgeschichte der Linux- und Open-Source-Einführung im Auswärtigen Amt

TORSTEN WERNER

(CC-Lizenz siehe Seite 563)

Das Auswärtige Amt stellt seit mittlerweile sechs Jahren erfolgreich seine IT–Infrastruktur zunehmend auf Open Source um und das bisher weitgehend unbeachtet durch die Öffentlichkeit. Der folgende Artikel zeigt die Geschichte der Open-Source-Einführung sowie deren aktuellen Stand. Dabei werden die Besonderheiten einer weltweiten Infrastruktur und spezifische Herausforderungen beschrieben, die es so nur in wenigen Institutionen der Welt gibt. Auch so genannte *global player* der Industrie haben keine vergleichbar große, globale Infrastruktur. Auf langwierige vorbereitende Studien wurde bewusst verzichtet. Vielmehr wurden immer pragmatische Lösungen gewählt. Zwei Punkte waren für die Migration besonders wichtig: einerseits die Einführung eines leistungsfähigen, zentralen Verzeichnisdienstes auf Basis von Open-Source-Werkzeugen und andererseits die frühzeitige sowie konsequente Umstellung auf plattformunabhängige Fachanwendungen.

Schlüsselwörter: Migration · Auswärtiges Amt · Verzeichnisdienst

1 Einleitung

Die Hauptaufgabe des Auswärtigen Amts ist, die deutsche Außenpolitik zu vertreten. IT ist dabei eine Verwaltungs- aber keine Kernaufgabe, welche mit einem vergleichsweise kleinen IT-Budget erledigt wird, was sowohl gilt, wenn es mit anderen Bundesressorts als auch mit den Außenministerien anderer westlicher Länder verglichen wird. Das ist sicher einer der Gründe für die relative Stille um die IT-Entwicklungen.

* Linus Torvalds (1999): „I'd like to say that I knew this would happen, that it's all part of the plan for world domination."

Das Amt unterhält neben der Zentrale in Berlin und Bonn auch 226 Auslandsvertretungen (Botschaften und Generalkonsulate) mit insgesamt ca. 11 000 IT-gestützten Arbeitsplätzen, wobei fast 4 000 Personen jährlich durch Rotation ihren Dienstposten in der Welt wechseln. Zusammen bilden sie den Auswärtigen Dienst, zu dessen Hauptzielen es gehört, den Frieden in der Welt mit diplomatischen Mitteln zu erhalten, den Aufbau eines vereinten Europas zu fördern sowie deutsche Interessen im Ausland zu vertreten. Dazu sind insbesondere moderne IT-gestützte Kommunikationsmittel erforderlich.

Das IT-Referat ist in der Zentrale angesiedelt und nicht auf mehrere Abteilungen verteilt, wobei es in New York und Singapur zusätzlich zwei kleine IT-Servicezentren gibt, welche die jeweiligen Zeitzonen betreuen. Eine Nacht gibt es im Auswärtigen Amt nicht und selbst das Wochenende ist kurz, weil einige Länder auch am Sonnabend arbeiten. Das IT-Referat besteht aus mehreren Fachbereichen, wobei dem Bereich IT-Strategie eine gewisse Führungsrolle zukommt. Dieser verwaltet den IT-Haushalt, hat die Verantwortung für die IT-Sicherheit und entscheidet wesentliche strategische Fragen. Für die Umsetzung sind jeweils andere IT-Fachbereiche zuständig. Die dem Amt zur Verfügung stehenden Mittel sind vergleichsweise gering. So verfügt das gesamte Amt nur über ca. 0,9 Prozent des Bundeshaushalts und das IT-Budget beträgt nur ein Viertel des IT-Haushalts der Vereinten Nationen bei einer ähnlichen Größe der Gesamtorganisation.

Heute geschieht die weltweite Kommunikation nicht mehr über analoge Funkverbindungen und auch nicht mehr nur über Papierpost und Telefon. Vielmehr unterhält das Auswärtige Amt ein über *IPsec*[1] geschütztes Intranet, auf das alle Beschäftigten weltweit und zeitgleich zugreifen können sowie darüber E-Mails und formalisierte Berichte und Erlasse austauschen. Vorgesehen ist außerdem die schrittweise Umstellung von papiergebundenen auf elektronische Abläufe (z. B. die Einführung einer elektronischen Registratur) und die Verringerung des Verwaltungsanteils durch optimale Nutzung aller technischen Möglichkeiten zur Datenverwaltung und Kommunikation.

2 Geschichte der Open-Source-Einführung

Bis zum Jahr 2001 war das Auswärtige Amt wie viele Behörden von proprietärer Software des Herstellers *Microsoft* dominiert. Im Wesentlichen waren sogar noch Produkte wie *Windows 3.11* im Einsatz mit dem heute kaum noch bekannten *Comfodesk* oder das Betriebssystem *Windows NT*. Das Mailsystem war *Microsoft Mail*, allerdings hatten die wenigsten Vertretungen eine elektronische Kommunikation mit der Zentrale. Wenn sie vorhanden war, wurde sie mit teuren *X.25-Schlüsselgeräten*[2] gesichert, wobei nicht nur die Geräte teuer waren, sondern auch die Übertragungskos-

1 *IPsec* steht für *Internet Protocol Security*, siehe auch http://de.wikipedia.org/wiki/IPsec.

2 *X.25* ist eine von der *ITU-T* standardisierte Protokollfamilie für großräumige Computernetze (WANs) über das Telefon-Netzwerk. Siehe auch http://de.wikipedia.org/wiki/X.25.

ten nach Volumen abgerechnet wurden und die damaligen Preise heute kaum noch vorstellbar sind. Die für eine vollständige sichere Vernetzung des Amtes benötigte Zeit wurde damals auf zehn bis zwölf Jahre geschätzt, wenn man weiter auf die herkömmlichen technischen Lösungen gesetzt hätte.

Im Anschluss an eine Reorganisation der IT-Referate wurde das so genannte VPN–Projekt im Jahr 2001 begonnen mit dem Ziel, bis Ende 2003 die Vollvernetzung der Zentrale mit allen Auslandsvertretungen zu erreichen unter Nutzung von modernen Internettechnologien wie *Virtual Private Network (VPN)*[3]. Aufgrund geringer finanzieller Mittel war der Einsatz von Windows-Servern ausgeschlossen, was nicht nur an den Lizenzkosten für *Windows* lag, sondern an den in Zusammenhang damit zwangsläufig anfallenden Kosten für Exchange-basierte Mailserver, Datenbankserver, Fernwartungssoftware usw. Ein Verzicht auf lokale Server und ein stattdessen direkter E-Mail-Abruf aus der Zentrale schied mangels breitbandiger und hochverfügbarer Netzanbindung aus. Die Server (ein bis zwei Stück pro Liegenschaft) wurden fast ausschließlich mit Open-Source-Software ausgestattet: *Red Hat*, *Exim*, *Squid*, *Samba*, *OpenLDAP*[4], *Webmin*[5] und *Apache*. Clientseitig kam *Windows XP* mit *Microsoft Office*, dem *Internet Explorer* und *Outlook Express* zum Einsatz. Auf das Produkt *Outlook* wurde bewusst verzichtet, um proprietäre Datenformate und Netzwerkprotokolle möglichst zu vermeiden. Zur Überwachung der Server und Netze dient *HP OpenView*. Zur Sicherung des Netzwerkes wird *IPsec* in Form der SINA-Technologie[6] verwendet, die im Auftrag des Bundesamts für Sicherheit in der Informationstechnik (BSI) von der *Secunet AG* entwickelt wurde.

Die Hochverfügbarkeit des VPN gründet sich einerseits auf einer sternförmigen Topologie mit zwei zentralen Standorten in Berlin und Bonn. Andererseits gibt es in jeder Vertretung einen Router, der zwischen zwei alternativen Internetprovidern umschalten kann und auch eine Lastverteilung der Bandbreite beherrscht. Außerdem sind die als *SINA-Boxen* bezeichneten IPsec-Gateways redundant ausgelegt. In den Jahren 2002 bis 2003 fand der Rollout der Server, Clients und Netzwerkkomponenten an allen Auslandsvertretungen statt. Seit dem Abschluss des VPN-Projekts steht dem Auswärtigen Dienst ein modernes weltweites Datennetz zur Verfügung. Es ist für die Übertragung von Verschlusssachen mit der Einstufung „nur für den Dienstgebrauch" geeignet und aufgrund einer besonderen nachrichtendienstlichen Gefährdung des Amts bis auf eine gut gefilterte E-Mail-Schnittstelle streng vom Internet getrennt.

3 Ein *Virtual Private Network (VPN)* (dt.: Virtuelles Privates Netz) ist ein Computernetz, das zum Transport privater Daten ein öffentliches Netz (zum Beispiel das Internet) nutzt. Siehe auch bei http://de.wikipedia.org/wiki/Virtual_Private_Network.

4 *OpenLDAP* ist eine Implementation des LDAP-Protokolls als freie Software. Es ist Bestandteil der meisten aktuellen Linux-Distributionen und läuft auch unter verschiedenen UNIX-Varianten, *Mac OS X* und verschiedenen Windows-Versionen. Die Webseite des Projekts findet sich unter http://www.openldap.org.

5 *Webmin* ist ein freies Programmpaket zur Verwaltung eines Rechners mit einem UNIX-artigen Betriebssystem aus der Ferne. Siehe auch unter http://www.webmin.com.

6 *Sichere Inter-Netzwerk Architektur*: http://www.bsi.bund.de/fachthem/sina/index.htm.

In der Zentrale dagegen blieb das dominierende Betriebssystem *Windows NT* auf Servern und Clients bis Mitte 2005 erhalten. Wesentliche sonstige Software war *Exchange 5.5*, *Outlook 97*, *Internet Explorer* und *Microsoft Office 97* – also eine typische IT-Umgebung in der öffentlichen Verwaltung. Erschwerend kam hinzu, dass als Netzwerktechnik *ATM*[7] über Lichtwellenleiter bis zum Arbeitsplatz eingesetzt wurde. Passende Hardware wurde immer unbezahlbarer. Im zentralen Intranet und auf so genannten Internet-Terminalservern gab es aber auch schon ab Anfang 2004 *Debian*[8]. Im Sommer 2005 fand dann das größte Migrationsprojekt statt: die Umstellung auf Ethernet, ausschließlich Debian-Server und Multiboot-Clients.

3 Aktueller Stand und strategische Überlegungen

Hardwareseitig werden Server von *Hewlett Packard* mit zwei bis vier Opteron–Prozessoren eingesetzt. Die Clienthardware kommt von *Fujitsu Siemens* mit Athlon64-Prozessoren und Notebooks von *IBM/Lenovo*. Als *Personal Digital Assistent (PDA)* werden in geringem Umfang der *Pocketloox*, ein *PocketPC*, eingesetzt, der in der Lage ist, sich mit der weltweit eingesetzten Groupware zu synchronisieren. Wesentliche Serversoftware ist *OpenLDAP*, *Samba 3* und ein Cyrus-2.2-Murder-Cluster. *Debian* wird in einem gemischten 32-/64-Bit-Modus betrieben, d. h. mit 64-bittigem Systemkern und 32-bittigem Benutzermodus. Neue Fachanwendungen werden webbasiert entweder mit *JBoss*[9], *Perl* oder *PHP* implementiert. Dabei liegt der Schwerpunkt auf *JBoss/Java*. *Exchange*, *Outlook* und *Internet Explorer* wurden durch die plattformunabhängigen Anwendungen *Thunderbird*, *Firefox* und die webbasierte Groupware *x-manage* ersetzt. Weitere wichtige plattformunabhängige Anwendungen sind der *VideoLAN client (VLC)* und *OpenOffice.org*[10], welches unter *Windows* zusätzlich zu *Microsoft Office XP* installiert ist. Neben Open Source wird auch besonders auf die Nutzung offener Standards, offener Netzwerkprotokolle und offener Dateiformate geachtet. Umso mehr zu begrüßen ist die Aufnahme des ISO-Standards *OpenDocument* in die *whitelist* des von der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung (KBSt) veröffentlichten SAGA-Dokumente (KBSt 2006).

7 *Asynchronous Transfer Mode (ATM)* ist eine Technik der Datenübertragung, bei der der Datenverkehr in kleine Pakete, Zellen genannt, mit fester Länge (53 Byte) codiert und über asynchrones Zeitmultiplexing übertragen wird. Siehe auch http://de.wikipedia.org/wiki/Asynchronous_Transfer_Mode.

8 *Debian* ist eine GNU/Linux-Distribution, die ausschließlich aus freier Software besteht. Die Webseite des Projekts ist zu finden unter http://www.debian.org.

9 Der *JBoss Application Server* ist die Implementierung eines Application-Servers nach dem J2EE-Standard und Teil des JBoss-Middleware-Frameworks. Siehe auch http://www.jboss.com.

10 *OpenOffice.org* ist ein freies Office-Paket, das aus einer Kombination verschiedener Programme zur Textverarbeitung, Tabellenkalkulation, Präsentation und zum Zeichnen besteht. Ein Datenbankprogramm und ein Formeleditor sind ebenfalls enthalten. Die deutschsprachige Webseite des Projekts befindet sich unter http://de.openoffice.org.

Das Auswärtige Amt hat sich Anfang 2004 für den Einsatz von *Debian GNU/Linux* als primäre Linux-Distribution für Server entschieden. Die Entscheidung wurde dann später folgerichtig auch auf Clients ausgeweitet. Gründe dafür waren u. a., dass *Debian* kein Außenseiter in der Linux-Welt ist und dass es herstellerunabhängig ist. Die konkurrierenden kommerziellen Distributionen sind weder frei noch preiswert. Zwar handelt es sich auch dabei um Open Source, aber Änderungen sind nicht erlaubt, ohne den gekauften Support einzubüßen. Damit verliert man einen der größten Vorteile von Open Source und damit einen der Gründe, weswegen Open Source zunächst eingeführt wurde. Der vermeintlich mangelnde Debian-Support hat sich nicht als Problem herausgestellt. Vielmehr findet man eine z. T. hervorragende Community und auch eine ganze Reihe von Dienstleistern mit Debian-Support, welche ebenfalls die Abhängigkeit von einem einzelnen Hersteller vermeidet. Kommerzielle Supportverträge werden gern nach dem „Tintenstrahl-Drucker-Modell" gestaltet, was meint, dass die Anschaffung eines Produktes zwar relativ günstig sein kann, aber dessen Folge- und Wartungskosten mangels eines tatsächlichen Wettbewerbs entsprechend hoch sind. Das betrifft aber nicht nur den speziellen Fall proprietärer Betriebssysteme, sondern alle Komponenten der eigenen IT-Infrastruktur wie Datenbanken, Fachanwendungen, so genannte „Standardsoftware", Speichersysteme, Datensicherung, Netzwerk usw. Bei der Entwicklung von Individualsoftware beispielsweise im Rahmen von Fachverfahren spielt das Betriebssystem keine Rolle und *Debian* wird von vielen Softwarefirmen unterstützt. Ein weiterer wichtiger Grund für den Open–Source-Einsatz ist deren immanent höhere Sicherheit.

Eine Besonderheit von *Debian* gegenüber „herkömmlichen" Softwareanbietern ist die Tatsache, dass es keine scharfe Trennung zwischen Entwicklern und Anwendern gibt. Debian-Entwickler programmieren meist für den Eigenbedarf, den Bedarf von Arbeitgebern, Universitäten, Schulen oder auch Verwandten und Freunden (Robles et al. 2005). Umgekehrt kann jeder Anwender auch direkt an *Debian* mitarbeiten und das schon seit Anfang an. Möglichkeiten der Mitarbeit gibt es sehr viele, wie beispielsweise über das Bug-Tracking-System, das Debian-Translation-Projekt, durch Bewerbung als neuer Entwickler oder auch durch Mitarbeit an einem der immer zahlreicher werdenden Pakete mit einem offenen, kollaborativen Entwicklungsansatz. Trotz alledem haben die Debian-Entwickler die Wünsche der Endanwender immer stark im Fokus, was *Debian* von anderen eher an Experten gerichteten Betriebssystemen wie *Gentoo* oder die freien BSD-Varianten unterscheidet.

Der zu einem nicht unwesentlichen Teil selbst entwickelte Multiboot-Client umfasst vier Systeme: zunächst die beiden Hauptsysteme *Windows XP* und *Debian*. Letzteres funktioniert trotz kurzer Entwicklungszeit teilweise besser als *Windows*, wobei der Entwicklungszeitraum relativ gering war im Vergleich zu Zeit und Aufwand, den Windows-Client für die besonderen Sicherheitsanforderungen zu härten. Weiterhin installiert ist ein so genanntes „global roaming" basierend auf *Debian*, welches ein Anmelden an jedem Arbeitsplatz unabhängig von der Zugehörigkeit des eigenen Kontos

zu Windows-Anmeldedomänen erlaubt. Das vierte System ist ein *SINA Thin Client*, der zukünftig eine Bearbeitung von als vertraulich eingestuften Verschlusssachen am Arbeitsplatz ermöglichen soll unter Nutzung von *Security Token*, *IPsec* und speziell gesicherten Terminalservern.

Notebooks werden inzwischen nur noch mit *Debian* installiert, wobei hier eine leicht angepasste Variante namens *SINA Native Client* mit dem Codename „DebiAAn" verwendet wird, die in Abstimmung mit dem Bundesamt für Sicherheit in der Informationstechnik entwickelt wurde. Gerade jetzt während der deutschen EU–Ratspräsidentschaft finden diese Notebooks reißenden Absatz, was die Verbreitung von *Linux* sehr fördert und von den Nutzern wegen des großen Vorteils von mobilen Rechnern gern angenommen wird. Vorbehalte gegenüber der Linux-/KDE-Oberfläche sind praktisch nicht vorhanden. Die Nutzerakzeptanz wird weiterhin mit einem regelmäßig erscheinenden IT-Newsletter gefördert. Mit Hilfe von vielen speziell für die eigenen Bedürfnisse entwickelten Webmin-Module sowie mit dem *Open-Source–Ticket-Request-System (OTRS)*[11] wird die Effizienz der IT-Administration gesteigert und der Anwender bei Problemen besser betreut. Das hilft auch, bei einer lebendigen IT-Landschaft die Übersicht über die eigenen Systeme nicht zu verlieren. Wichtig ist weiterhin, dass trotz häufiger Innovationen und Migrationen die Nutzer weitgehend ungestört arbeiten können.

Fachverfahren werden schon seit einigen Jahren nur noch als Webanwendungen realisiert, die üblicherweise schon vom Design her plattformunabhängig sind. Außerdem vereinfachen sie die Softwareverteilung, weil sie nur auf Servern installiert werden und am Client keine oder nur geringe Änderungen erfordern. Hinzu kommt, dass ein Webformular sich auch leichter modifizieren lässt als eine typische Clientanwendung, so dass auch weniger erfahrene Entwickler produktiv mitarbeiten können. Durch Vorgabe von genauen technischen Spezifikationen gegenüber externen Softwareherstellern kann eine gute Interoperabilität zwischen den Anwendungen sichergestellt werden, so dass sich auch eine Vielzahl von Einzelanwendungen auf einem Server betreiben lassen. Letzteres ist gerade an den Auslandsvertretungen wichtig, wo nur relativ wenig Serverhardware direkt vor Ort installiert werden kann.

Geeignetes IT-Personal ist eine wichtige Voraussetzung für eine erfolgreiche Open–Source-Migration. Das beginnt schon mit der Einstellung von neuen Mitarbeitern: So hat das Amt zwei Debian-Entwickler und einen OTRS-Entwickler eingestellt. Bereits vorhandenes Personal muss natürlich geschult werden. Dabei haben sich die sehr beliebten Schulungen im *Linuxhotel*[12] in Essen bewährt, die man beinahe als „Gehirnwäsche" bezeichnen kann. Bewährt haben sich außerdem das partielle „Insourcing" von Kernkompetenzen wie bei unserem Basisserver und dem Debian-Client sowie die referatsübergreifende Zusammenarbeit in Projekten. Eine weitere angeneh-

11 *Open Ticket Request System (OTRS)* ist ein Open-Source-Trouble-Ticket-System. Die Homepage des Projekts findet sich unter http://otrs.org.

12 Siehe auch http://www.linuxhotel.de.

me Erfahrung ist, dass Open Source die Kreativität der Mitarbeiter erheblich steigern kann, weil Änderungen an der Software sehr viel weitgehender möglich sind, was zu entsprechend großen Erfolgserlebnissen führt (DiBona et al. 1999).

4 OpenLDAP als zentraler Verzeichnisdienst

OpenLDAP wird im Auswärtigen Amt an vielen Stellen eingesetzt. Zunächst ist es der führende Verzeichnisdienst für die Identifikation und Authentifikation von Benutzern unter Verwendung eines Rollenmodells. Das bedeutet, dass Computerkonten flexibel zu Personen zugeordnet werden können, je nach Dienstposten, den eine Person für eine bestimmte Zeit bekleidet. Aufgrund der regelmäßigen Rotation der Mitarbeiter ist eine solche Flexibilität unbedingt notwendig. Weiterhin dient der Verzeichnisdienst als Datenbank und elektronisches Nachschlagewerk beispielsweise für Telefonnummern, E-Mail-Adressen oder auch Personaldaten. Je nach Sensibilität der Daten können diese entweder von allen eingesehen werden oder sind nur von bestimmten Anwendungen les- und veränderbar. Ein weiteres Einsatzgebiet des Verzeichnisdienstes ist die Autorisierung, d. h. die Zuordnung von Berechtigungen zu Konten beispielsweise für den Zugriff auf Peripheriegeräte und auf Anwendungen, aber auch detaillierte Einzelberechtigungen innerhalb von Anwendungen. Hierzu gehört ebenfalls die Zuweisung von Speicherbegrenzungen (*quotas*) auf E-Mail-Postfächer und Netzlaufwerke sowie die Erlaubnis zum Farbdruck. Letzte wichtige Anwendung des Verzeichnisdienstes ist die elektronische Abbildung der Organisationsstruktur des Amts. Diese beinhaltet die verschiedenen Hierarchien wie Fachbereiche, Referate, Abteilungen und Auslandsvertretungen sowie die Zuordnung von Mitarbeitern zu den jeweiligen Vorgesetzten und Vertretern. Diese Informationen werden benötigt für die Abbildung von Arbeitsabläufen in elektronischen Workflowsystemen, zur Regelung von Zugriffsrechten auf elektronische Akten bis hin zur Strukturierung des internen Content-Management–Systems – dem *Intranet*. Aus Sicherheitsgründen und zwecks Ressourcenschonung ist der Verzeichnisdienst so aufgebaut, dass nur die benötigten Teilzweige an die Auslandsvertretungen repliziert werden. Die Daten selbst werden vorwiegend über individuell angepasste Webmin-Module gepflegt, wobei die Berechtigungen dafür ebenfalls im Verzeichnisdienst gespeichert werden.

Relativ neu, also weniger als drei Jahre jung, ist die Verwaltung von Konfigurationsdaten für Debian-Server und -Clients auf einem eigenen zentralen LDAP-Server. Das ermöglicht eine vollautomatische Installation vieler gleichartiger oder funktionsähnlicher Rechner auf einfache Art und Weise und auch die schnelle Neuinstallation eines Geräts bei Hardwaredefekten, wobei der Ursprungszustand völlig identisch reproduziert wird. Dabei wird ausgenutzt, dass die unter *Linux* vorwiegend im einfachen Textformat vorliegenden Konfigurationsdateien aus *Templates* durch Ausfüllen mittels der LDAP-Daten bzw. mit Standardwerten bei fehlenden LDAP-Einträgen einfach erzeugt werden können. Realisiert wird das mittels eigener Debian-Pakete,

die zur leichteren Erkennung die Namenskonvention „aa-$name.deb“ einhalten. Die Quelltexte werden in verschiedenen *Subversion repositories* gespeichert, so dass über die Versionsverwaltung auch gleich eine automatische Dokumentation des Entwicklungsstands erreicht wird. Auf proprietäre Lösungen wurde hier bewusst verzichtet, vielmehr ist auch der Mechanismus der Softwareverteilung mittels „absint“ (apt-basierte Softwareinstallation) sehr Debian-artig und wurde mit Hilfe von *Bash*, *Perl* und *Python* implementiert. Das ermöglicht die Wiederverwendung existierender Open–Source-Lösungen, Teile davon wurden aber auch selbst oder von externen Dienstleistern entwickelt. Einige interessante Funktionen sind die vollautomatische Installation von *Linux-HA-Clustern*[13], die sowohl hochverfügbar sind (Failover-Konfiguration) als auch in einer Aktiv-Aktiv-Konfiguration laufen (so genanntes *load balancing*), außerdem die vollautomatische Unterstützung von verteilten *DRBD volumes*[14], die mit dem *Logical Volume Manager (LVM)* flexibel verwaltet werden. Updates können dabei gleichzeitig auf vielen Rechnern verteilt und überwacht werden.

5 OTRS als Beispiel für eine erfolgreiche Fachanwendung

2004 standen die IT-Referate vor dem Problem, dass der eigene *User Help Desk (UHD)* mit einer unvollkommenen Anwendung arbeiten musste. Nachteile der proprietären Anwendung waren u. a., dass es zu wenige, weil teure Lizenzen für potenziell zu viele Anwender gab, dass die Software mangels E-Mail-Schnittstelle einen Medienbruch darstellte, dass sie schwer anpassbar war und eine langsame und nicht benutzerfreundliche *GUI* hatte. Plattformunabhängig war sie ebenfalls nicht, was zur Herstellerabhängigkeit führt, und außerdem beherrschte sie nur eine interne Authentifizierung der Agenten. Fast zeitgleich hatten die Kollegen vom Bürgerservice ein ähnliches Problem. Sie waren mit der notdürftig über *Outlook* gezimmerten Form der Bearbeitung von Bürgeranfragen nicht zufrieden, insbesondere wenn es immer mal wieder zu großen Wellen von Anfragen kommt.

Nach einer einwöchigen Schulung zu *OTRS*, einer Webanwendung speziell zur Unterstützung von *User Help Desks*, wurde diese Open-Source-Software getestet und bald auch eingeführt. Vorteile waren, dass mangels Lizenzkosten viel mehr Anwender bzw. Agenten die Software nutzen können. Weiterhin ist man nicht vom Hersteller abhängig, sondern kann entweder selbst Änderungen vornehmen oder durch Dritte vornehmen lassen. Die Software ist auch ausgesprochen leicht anpassbar, so dass sie für vom Hersteller nicht beabsichtigte Zwecke genutzt werden kann. Speziell im Fall des Bürgerservice war es wichtig, in Krisenfällen schnell weitere Agenten rekrutieren zu können, die keine separate Schulung benötigen, sondern sofort arbeitsfähig sind. Nur eine Software mit weitgehend konfigurierbarer Benutzerschnittstelle ist hierfür geeignet.

13 Siehe *The High-Availability Linux Project* unter http://www.linux-ha.org.
14 Siehe *Distributed Remote Block Device* unter http://www.drbd.org.

Heute wird *OTRS* auch in vielen weiteren Arbeitseinheiten eingesetzt. So war das Auswärtige Amt das erste Bundesministerium, das Anfragen nach dem am 1. Januar 2006 eingeführten Informationsfreiheitsgesetz mit einer IT-Anwendung bearbeitete, konkret natürlich mit *OTRS*, und diese daher besonders effizient abarbeiten kann. Hierbei übernimmt *OTRS* die Aufgabe eines elektronischen Vorgangsbearbeitungssystems und vermeidet die Einführung proprietärer und unflexibler Produkte. Insgesamt ist *OTRS* ein sehr gutes Beispiel für eine Webanwendung, die sich sehr gut an Fachanforderungen anpassen lässt und damit IT-gestützte Fachverfahren auf Open-Source-Basis ermöglicht.

6 Pläne

Zukunftspläne sind die Ablösung alter Fachverfahren durch moderne, elektronische Webanwendungen. Dabei kann es sich durchaus auch um Abläufe handeln, die bisher noch gar nicht elektronisch unterstützt werden oder wo das aktuelle IT-Verfahren entweder ineffizient oder nicht plattformunabhängig ist. Hierzu gehören unter anderem Personalanwendungen, digitale Workflows unter Verwendung von *jBPM*[15] und *OpenOffice.org*, eine „elektronische Registratur" als Dokumentenmanagementsystem sowie die politisch sehr forcierten Biometrieanwendungen für das Ausstellen von Pässen und Visa. Andere Clientanwendungen werden zunehmend durch plattformunabhängige Software, Open Source und *Debian* ergänzt und/oder ersetzt. Aus Sicherheitsgründen wird *Windows* immer mehr nur noch in einem virtuellen Container eingesetzt mit Einschränkungen wie einem nichtpersistenten Dateisystem und sehr stark kontrolliertem Netzzugang, um Angriffe von Schadsoftware in Grenzen zu halten. Weiterhin werden immer mehr Debian-basierte Terminalserver eingerichtet, um einen sicheren Internetzugang an jedem Arbeitsplatz zu ermöglichen. *PostgreSQL*[16] wird bald das führende relationale Datenbankmanagementsystem sein. Das betrifft beispielsweise auch die Migration des vom BundOnline-Projekt des Bundesinnenministeriums bereitgestellte Content-Management-System nach *Debian*, *PostgreSQL* und *Firefox*.

Weiter verstärkt wird die Rückgabe von für das Auswärtige Amt oder in diesem entwickelter Software an die Open-Source-Community, was aber leider aufwändiger ist, als es auf den ersten Blick erscheint. Die Schwierigkeiten liegen in der starken Integration der IT-Systeme untereinander, so dass sich eine Open-Source-Komponente schwer extrahieren lässt, ohne deren Funktionstüchtigkeit erheblich einzuschränken. Oft stellt man auch fest, dass es keine saubere Trennung von Konfigurationsdaten und spezifisch angepasstem Programmcode von allgemein verwendbarem Code gibt. Am einfachsten ist die Weitergabe von Patches für Basiskomponenten an die jeweiligen

15 Siehe auch http://www.jboss.com/products/jbpm.

16 *PostgreSQL* ist eines der ältesten und am weitesten fortgeschrittenen objektrelationalen Datenbanksysteme (ORDBMS) im Bereich freie Software. Siehe Projekt-Webseite unter http://www.postgresql.org.

Softwareautoren. Schließlich soll die Kooperation mit anderen Behörden intensiviert werden, die bereits Open Source einsetzen und an einer solchen Kooperation interessiert sind.

7 Fazit

Das Beispiel der Linux-Migration des Auswärtigen Amts zeigt, dass es auch in einer größeren und geografisch weit verteilten Institution möglich ist, in verhältnismäßig kurzer Zeit eine auf Open Source basierende IT-Infrastruktur aufzubauen. Besondere Vorteile dieser Umstellung sind die höhere Wirtschaftlichkeit, bessere Interoperabilität und größere Sicherheit der IT-Systeme sowie hoch motivierte IT-Mitarbeiter. Taktisch geschicktes Vorgehen vorausgesetzt, greift die Begeisterung auf die Endanwender über, was die Einführung von *Linux* und Open Source auf dem Desktop erheblich vereinfacht. Wesentlich ist dabei, alle Facetten und Aspekte der eigenen IT-Infrastruktur im Auge zu haben und in einer konzertierten, aber nicht überhasteten Aktion die Migration Schritt für Schritt voranzutreiben und bei unvermeidlich auftretenden Problemen nicht zu verzagen.

Literatur

DiBona, C., Ockman, S. und Stone, M. (Hrsg.) (1999), *Open Sources: Voices from the Open Source Revolution*, 1. Aufl., O'Reilly, Cambridge, MA, USA. http://www.oreilly.de/catalog/opensources/book/toc.html [22. Jan 2007].

KBSt (2006), 'Standards und Architekturen für E-Government-Anwendungen'. http://www.kbst.bund.de/saga [10. Jan 2007].

Robles, G., Gonzalez-Barahona, J. M. und Michlmayr, M. (2005), Evolution of Volunteer Participation in Libre Software Projects: Evidence from Debian, *in* M. Scotto und G. Succi (Hrsg.), 'Proceedings of the First International Conference on Open Source Systems', Genua, S. 100–107. http://www.cyrius.com/publications/robles_barahona_michlmayr-evolution_participation.html [22. Jan 2007].

Torvalds, L. (1999), The Linux Edge, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices from the Open Source Revolution', 1. Aufl., O'Reilly, Cambridge, MA, USA. http://www.oreilly.com/catalog/opensources/book/linus.html [23. Jan 2007].

Nach der Migration – business as usual?

CARSTEN BRUNKE UND JÖRG MEYER

(CC-Lizenz siehe Seite 563)

Bei der Norddeutschen Affinerie AG (NA) wurde im Jahr 2004 Novell- durch Linux-Server ersetzt. Im Open Source Jahrbuch 2005 berichteten wir darüber.[1] Wie ging es seitdem weiter? Hat sich die Umstellung bewährt? Ist die Migration auch nach knapp drei Jahren Betrieb noch die finanziell bessere Wahl? Welche Entwicklung nahm die IT-Infrastruktur der Norddeutschen Affinerie AG im Hinblick auf Open-Source-Software? Welche Pläne wurden verwirklicht, welche kamen hinzu? Diese Fragen beantwortet der folgende Artikel.

Schlüsselwörter: Migration · Samba · OpenLDAP · NX NoMachine

1 Was vorher geschah – die Revolution

Im März 2003 stand die Norddeutsche Affinerie AG (NA), Europas größter Kupferhersteller und weltgrößter Kupferrecycler, vor der Entscheidung, möglicherweise wichtige Teile der IT-Infrastruktur von proprietären Produkten auf freie Software umzustellen. Die IT-Leitung der NA 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*[2] und *OpenLDAP*[3] in Zusammenarbeit mit einem externen strategischen Partner, der *inmedias.it GmbH*. Seit Ostern 2004 ist die neue

1 Siehe Meyer und Brunke (2005).

2 Samba ist eine Open-Source-Software (OSS), mit der in einem Netzwerk Datei- und Druckdienste zur Verfügung gestellt werden. Diese Dienste können sowohl von Linux-Clients als auch von Windows-Clients genutzt werden. Weitere Informationen finden sich unter: http://samba.sernet.de/samba.7.html.

3 *OpenLDAP* ist eine OSS-basierte Implementierung des *Lightweight Directory Access Protocol (LDAP)* und stellt somit einen Verzeichnisdienst (in einem Verzeichnis können Daten unterschiedlicher Art in einer festgelegten Struktur gespeichert werden) zur Verfügung. Informationen zu *OpenLDAP* finden sich unter: http://www.openldap.org.

Struktur live und erreicht eine Verfügbarkeit von 99,7 Prozent, basierend auf einem Linux-Server mit Standardhardware.

Statt eingeplanter neunzig Tage *Troubleshooting*[4] 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 mit der Migration der Systeme.[5] Zudem wurde durch nicht mehr anfallende Lizenzkosten das Ziel einer spürbaren spontanen Kostensenkung erreicht. Weitere Forderungen an das neue System umfassten eine deutliche Erhöhung der Verfügbarkeit, sinkende Administrationsaufwände und die Eignung als Basis für eine Ausweitung des Einsatzes von OSS – dieser Artikel berichtet, wie es um das Verhältnis von Ist und Soll bestellt ist.

Im September 2004 wurde das beschriebene Migrationsprojekt mit dem dritten Platz des *Open Source Best Practice Awards*[6] prämiert.

2 Was seither geschah – die Evolution

Dem ersten Migrationsschritt – der Revolution – folgte aufgrund der guten Erfahrungen schnell die evolutionäre Phase. Diese umfasst Homogenisierung (verbunden mit der Ausbreitung von *Linux* auf zusätzliche Konzernstandorte) und Stabilisierung des IT-Betriebs bei niedrigen laufenden Kosten, Ausweitung der Einsatzgebiete der eingesetzten OSS, Funktionserweiterungen und Einführung weiterer Linux-basierter Software.

2.1 Homogenisierung und Stabilisierung

Ein wesentlicher Aspekt der Strategie der NA war von Anfang an die Nutzung der integrativen Fähigkeiten von Open-Source-Software. Bedingt durch zahlreiche Firmenzukäufe gingen „über Nacht" häufig komplette IT-Infrastrukturen in die Verantwortung der Konzern-IT über. Typischerweise handelt es sich hierbei um höchst heterogene und in der Mehrzahl Microsoft-dominierte IT-Landschaften. Die konzeptionell beabsichtigte Zusammenarbeit insbesondere von *Samba* und *OpenLDAP* mit diesen Welten bedeutet einigen Spielraum bei der Handhabung der zu integrierenden Firmennetzwerke.

Ein Beispiel: Im Jahr 2005 wurde ein Hüttenwerk im Westen der Bundesrepublik Deutschland von der NA übernommen. Zweihundert Windows-Arbeitsplätze befanden sich in einer Windows-NT-4.0-Domänenstruktur mit mehreren Servern.

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

5 Tatsächlich macht der Zuwachs an Know-how und die Beschäftigung mit dem neuen System auf unterster Ebene schlicht Spaß.

6 Zum 2004 erstmalig vergebenen Award, der jeweils in den Kategorien „Freie Wirtschaft" und „Öffentliche Verwaltung" vergeben wird, finden sich hier nähere Informationen: http://soss.lightwerk.com/content/award/index_ger.html.

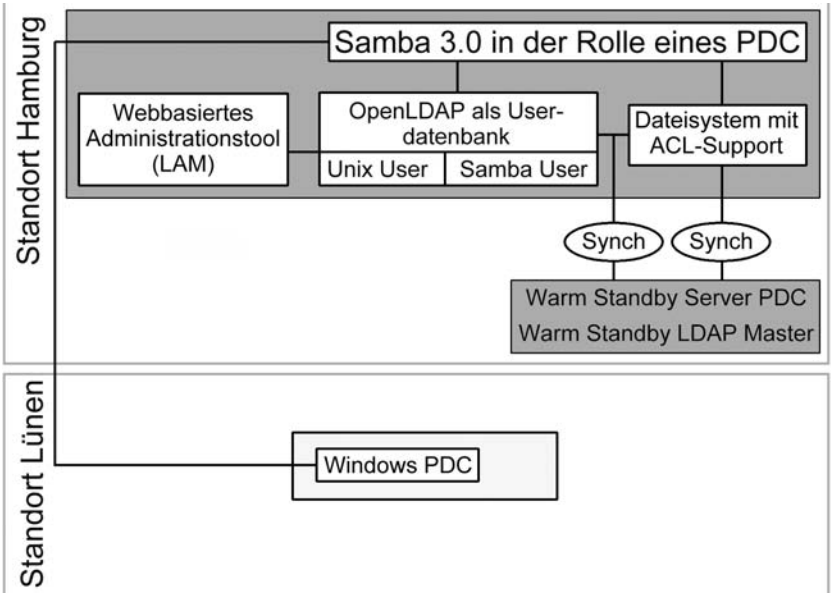


Abbildung 1: Vertrauensstellung

Eine schnelle Maßnahme zur Einbindung dieser Struktur in das Linux-Backoffice[7] der Firmenzentrale bestand in der Einrichtung einer Vertrauensstellung zwischen den Domänen (siehe Abbildung 1). Bei diesem Konstrukt betrachten Windows-NT–Server Nutzer und Ressourcen anderer Windows-NT-Server als die eigenen – dass in diesem Fall die „anderen Windows-NT-Server" durch Samba-Server unter *Linux* repräsentiert wurden, tat der Funktion keinerlei Abbruch. Von einem weniger technischen Standpunkt aus lässt sich auch formulieren, dass der Windows-NT-Server den Linux-Server als „einen von uns" akzeptiert.

Nachdem mit diesem Schritt die Wartbarkeit des Systems hergestellt wurde, folgte die kurze Planungsphase für eine komplette Migration des neuen Standorts zu *Samba* und *OpenLDAP*. Wenige Wochen später löste ein Linux-Server die Windows-Systeme im Backoffice ab (siehe Abbildung 2). Hierbei wurde der erste Schritt für eine Nutzung des Verzeichnisdienstes *OpenLDAP* als firmenweites, zentrales Identitätsmanagementwerkzeug getan.

Zu diesem Zweck wurden zwei so genannte OpenLDAP-Slaves aufgesetzt, deren Aufgabe in einer Spiegelung des Datenbestandes des Verzeichnisdienstes über die Unternehmensstandorte hinweg besteht. Ziel der Maßnahme ist einerseits eine

7 Unter dem Begriff *Backoffice* sind sämtliche Dienste zusammengefasst, die von einem Verbund von Servern zentral, im „Backoffice", zur Verfügung gestellt werden.

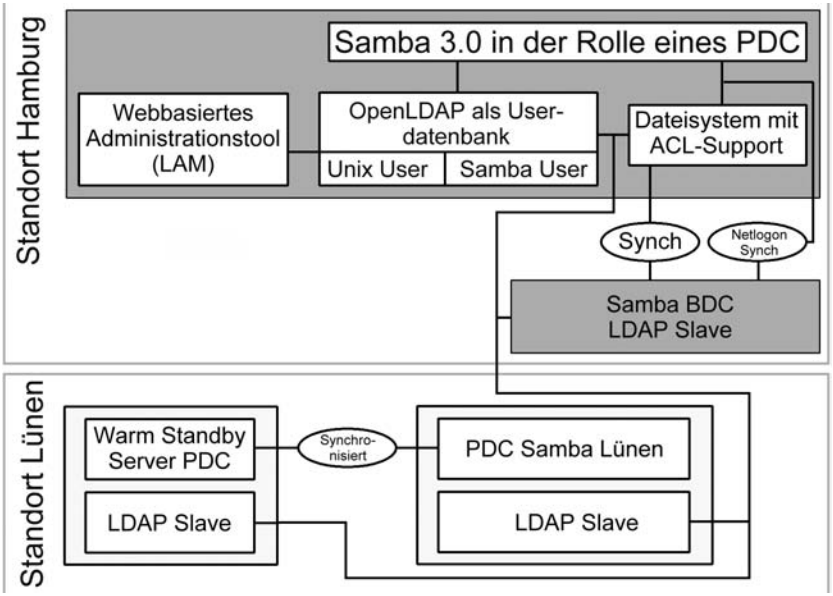


Abbildung 2: Einbindung entfernter Standorte

Verteilung der Aufgabe „Netzwerkanmeldung" über mehrere Geräte. Andererseits bedeutet der Einsatz von OpenLDAP-Slaves die Erhöhung der Verfügbarkeit des Anmeldedienstes – bei einer Unterbrechung der Verbindung zur und selbst bei einem Ausfall des Master-Verzeichnisservers in der Konzernzentrale.

Ein längerer Ausfall des Masters bedeutet allerdings Funktionseinschränkungen. So können keine Änderungen an den Verzeichnisdaten vorgenommen werden, da nur der Master Schreibzugriffe auf den Datenbestand hat. Diese Sollbruchstelle kommt durch eine Einschränkung der freien Software *OpenLDAP* zustande – die fehlende Fähigkeit wird als „Multi-Master-Betrieb" bezeichnet.[8] Ein Teil des Rezeptes des andauernden und durchgängigen Erfolges des Einsatzes von *Linux* ist die Einbeziehung des IT-Personals der NA in die Projektdurchführung durch das dienstleistende Unternehmen. Aufgrund des stets stattfindenden Know-how-Transfers konnte auf diese Weise, ebenfalls im Jahr 2005, die Infrastruktur zweier kleinerer Tochterfirmen komplett ohne externe Unterstützung auf *Linux* und *Samba* umgestellt werden.

Auch ein Blick auf die Zahlen zur Verfügbarkeit des Gesamtsystems bestätigt, insbesondere auch im Rückblick auf die Werte aus der Zeit vor der initialen Migration,

8 Zum Zeitpunkt der Erstellung dieses Artikels wird an der Beseitigung dieser Funktionslücke konzeptionell gearbeitet. Zu den möglichen Lösungswegen zählen der Einsatz einer kommerziellen OpenLDAP–Multi-Master-Erweiterung und das Setup von *OpenLDAP* in einem Cluster mit zentralisierter Datenhaltung.

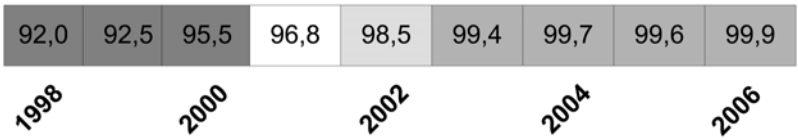


*Abbildung 3: Durchschnittliche Verfügbarkeit in % (365d*24h)*

eine erfreuliche Tendenz. Trotz eines stetigen Anstiegs der Komplexität der Systeme ergibt sich zum Zeitpunkt des Entstehens dieses Artikels für die ersten zehn Monate des Jahres 2006 eine Verfügbarkeit von 99,9 Prozent (siehe Abbildung 3).

Es bleibt an dieser Stelle einen Blick auf die Erfahrung der NA mit der Kostenseite einer Migration zu *Linux* im Backoffice zu werfen. Eine Offenlegung der genauen Zahlen ist leider nicht möglich, aber einige interessante Eckdaten können hier einen kleinen Einblick geben: Während die Kosten für externe Beratung von 2004 bis 2006 um 50 Prozent gestiegen sind, fielen die laufenden Kosten für die gesamte Systempflege inklusive internem Management und Downtime-Bewertung um 7,8 Prozent geringer aus. In einer allerersten Näherung könnte hieraus die These abgeleitet werden, dass die Einsparungen im Lizenzkostenbereich keineswegs durch hohe Kosten für Schulungen und externes Know-how aufgezehrt werden. Vielmehr ist nach dem Eintreten des *ROI* nach 0,7 Jahren[9] auch im Bereich der laufenden Kosten eine signifikante Ersparnis zu beobachten. Insofern diese Beobachtung auf Daten aus nunmehr knapp drei Jahren Betrieb basiert, ist von einer entsprechenden Fortentwicklung auch für die Zukunft auszugehen.

2.2 Ausweitung der Einsatzgebiete

Die OpenLDAP-Implementierung bei der NA erschien Anfang des Jahres 2005 als hinreichend belastbare Basis, um über eine Ausweitung deren Einsatzes nachzudenken. Gefordert waren dabei der Aufbau einer zentralen Adressdatenbank für die interne Verwaltung von E-Mail-Adressen, Telefonnummern (Festnetz sowie mobil), Personal-Organisationszuordnungen etc.

Strategisches Ziel bei der Schaffung der zentralen Adressdatenbank war ein System, welches auf offenen Standards basiert. Das System sollte quelloffen sein und eine API zur Anbindung verschiedener Applikationen mitbringen. Über Standortgrenzen hinweg war performanter und sicherer Betrieb zu gewährleisten – namentlich kam es auf Skalierbarkeit und die Fähigkeit zur Replikation oder Clusterung zur Erhöhung der Ausfallsicherheit an. Zu guter Letzt bedurfte es auch einer Exit-Strategie, d. h. der Migrationsfähigkeit zu anderen Repositories innerhalb der nächsten fünf Jahre.

9 Siehe Meyer und Brunke (2005). *ROI = Return On Investment*, Amortisationszeit.

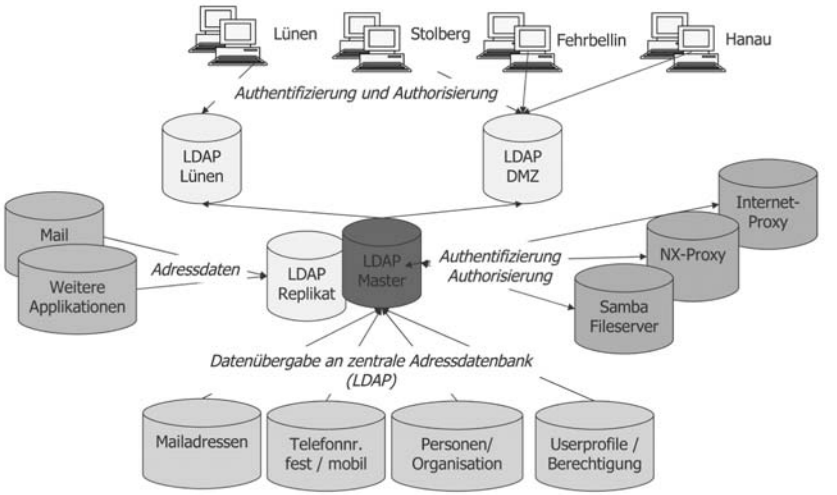


Abbildung 4: Zentrales identity repository

OpenLDAP bedient diese strategischen Ziele vollumfänglich und erfüllt zugleich die technischen Anforderungen an ein zentrales *identity repository* bei der NA:

- Automatischer Abgleich der zentralen Adressdatenbank bei Veränderungen in den Vorsystemen (*Sendmail*, *Oracle*).
- Zentraler Zugriff auf freigegebene Felder der zentralen Adressdatenbank für verschiedene Applikationen.
- Zentrale Stelle für Authentifizierung und Authorisierung der Fileserver-Dienste.
- Zentrale Stelle für Authentifizierung und Authorisierung der Terminalserver–Proxy-Dienste.
- Zentrale Stelle für Authentifizierung und Authorisierung der Internet-Proxy–Dienste.

Im konsolidierten *identity repository* (siehe Abb. 4) werden heute 15 000 LDAP–Objekte geführt und eine Vielzahl von Frontendsystemen bedient. Zur Administration werden der *LDAP Account Manager* (siehe folgender Abschnitt) und mehrere individuell entwickelte Java-Applikationen eingesetzt.

2.3 Funktionserweiterungen

Mit wachsender Bedeutung von *OpenLDAP* als Tool für das Indentitätsmanagement stiegen bei der NA auch die Anforderungen an die Verwaltungswerkzeuge. Von Be-

ginn an kam hierfür vorrangig der *LDAP Account Manager (LAM)*[10] zum Einsatz. Für diesen Einsatz war es erforderlich, *LAM* die eindeutige Identifizierung von Benutzern, Gruppen und Host-IDs über mehrere *LDAP container* hinweg beizubringen. Die Beschreibung der erfolgten Änderungen gerät sehr technisch und ist für das Verständnis aus Managementsicht nicht erforderlich. Gleichwohl ist sie aber vielleicht für manchen Leser mit ähnlichen Aufgabenstellungen von besonderem Interesse.

Aufgrund der Standortverteilung bei der NA werden verschiedene Container zur Verwaltung im Prinzip gleichwertiger Daten verwendet. Beispielsweise wird beim Anlegen eines neuen Benutzers für die Vergabe der neuen *UserID (UID)* nur der *parent container* durchsucht. Im Fall der NA führt das zu einem Kollisionsproblem, da die *UIDs* in allen Containern eindeutig sein müssen. Mit der bisherigen Berechnungsimplementierung war das nicht gewährleistet. Analog verhält es sich zu Gruppen- und Hostnamen.

LAM beschleunigt den Zugriff durch Zwischenspeicherung von LDAP-Daten in einer Webserversession. Mit diesen gespeicherten Daten wird beim Neuanlegen von Usern, Gruppen und Hosts die höchste existierende ID ermittelt und zum Anlegen um Eins inkrementiert. Mit der Erweiterung können in der LAM-Konfigurationsdatei beliebig viele weitere Container für diese Berechnung eingetragen werden. Diese werden ebenso in der Session zwischengespeichert und für die ID-Ermittlung mit einbezogen. So ist eine eindeutige Vergabe möglich. In einem weiteren Erweiterungsschritt wurde dieses Verfahren auch auf die Benutzernamen und die Gruppen ausgeweitet.

2.4 Einführung weiterer Linux-basierter Software

Neben der erfolgreichen Einführung von *OpenLDAP* und *Samba* wurden bei der NA weitere Linux-basierte Softwarelösungen eingeführt. Beispielhaft soll an dieser Stelle auf die Programme *SpamAssassin* und *NX NoMachine* eingegangen werden.

SpamAssassin

Als Mailserver kam bei der NA schon vor der initialen Migration ein kommerzieller Mailserver auf der Basis von *Sendmail* und *Linux* zum Einsatz. Daher bot es sich an, bei der Bekämpfung von Spam auf das freie Projekt *SpamAssassin* zu setzen. Die Inbetriebnahme und das tägliche Handling entpuppten sich als unproblematisch. Dennoch ist *SpamAssassin* eines der wenigen Themen, die nicht zur Erfolgsgeschichte wurden. Die Reaktion auf veränderte Strategien der Spammer erfolgte stets zu langsam. Im Ergebnis waren schubweise große Mengen unerkannter Spammails und leider auch nicht wenige *false positives*[11] zu beklagen. *SpamAssassin* wurde in der Zwischenzeit durch ein kommerzielles Produkt ersetzt.

10 http://lam.sourceforge.net

11 Als *false positives* werden in diesem Zusammenhang fehlerhaft erkannte Spammails bezeichnet, bei denen die Nachricht im Gegensatz zur Annahme des Systems keineswegs Spam ist.

NX NoMachine

Die regional verteilte Struktur der NA bringt es mit sich, dass auch Teile der Infrastruktur des einen Standorts an einem anderen genutzt werden sollen. So war bei der Einführung von *NX NoMachine*[12] gegeben, dass eine OLAP-Software auf einem (Windows-)Server in Hamburg auf (Windows-)Clients in Lünen genutzt werden sollte. Gegen die Nutzung von Windows-Bordmitteln sprach die zu geringe Performance bei gleichzeitig zu hohem Bandbreitenverbrauch. Die Einführung einer Lösung wie *Citrix* schied von vornherein aufgrund der zu hohen Lizenzkosten aus, insbesondere aber wegen deren Bindung an die Windows-Plattform.

Die Einführung von *NX* erfolgte nach einem Drehbuch, welches zum Lehrfilm über Projekte im Linux-Umfeld taugt. So folgte auf eine intensive Evaluierungsphase bei NA und Dienstleistern die Kontaktaufnahme zum Hersteller wegen „fehlender" Features und Integrationsproblemen mit dem doch recht ausgefeilten Identitymanagement auf der Basis von *OpenLDAP*. Beispielsweise beherrschte der *NX Client* seinerzeit nur 256 Farben. Für die Mehrzahl der Anwendungen auf dem Desktop wäre dies auch ausreichend; jedoch nicht bei der zu verteilenden OLAP-Software, welche ihre Informationsangebote durch feine Farbnuancen strukturiert. In Zusammenarbeit mit Entwicklern des Herstellers und Mitgliedern der FreeNX-Community entstanden die notwendigen Erweiterungen, welche heute im Standardprodukt jedem Anwender zur Verfügung stehen. Telefonkonferenzen mit den über mehrere Kontinente verteilten Entwicklern ließen dies zudem in Rekordzeit entstehen.

NX NoMachine der italienischen Firma *NoMachine s.r.l.* arbeitet als Terminalserver polymorph mit Windows- und Linux-Clients und -Servern. Im konkreten Einsatzfall transportiert *NX* wie ein Proxy die Terminalsessions der Windows-Terminalserver zu den entfernten Windows-Clients. Alleinstellungsmerkmale der Lösung, von der es auch eine freie Variante namens *FreeNX* gibt, sind die extreme Bandbreitenschonung, hohe Performance sowie ausgezeichnete Stabilität und Skalierbarkeit.

Ähnlich bemerkenswert gestaltete sich die Einbindung von *NX* in das Identity–Management. Um die Nutzerakzeptanz der Lösung zu sichern, forderte die NA ein effektives Single-Sign-On[13]. In der Standardlösung war es jedoch erforderlich, dass nach der Anmeldung des *NX Client* am *NX Server* zudem innerhalb der Session eine Anmeldung am darin transportierten Windows-Terminalserver erfolgte. Weniger technisch versierte Anwender verlieren in solchen Schachtelkonstruktionen schnell die Übersicht und damit die Motivation. Wiederum wurde in Zusammenarbeit mit

12 Laut *Wikipedia* ist *NX* „eine Remote-Desktop-Software der italienischen Firma *NoMachine*, die mit Ausnahme der zentralen Client- und Serverapplikation unter der *GNU GPL* steht. Mit *NX* kann man den Bildschirminhalt eines entfernten Computers auf einen lokalen Rechner (auch betriebssystemübergreifend) übertragen und damit arbeiten, als säße man direkt davor."

13 Der Anwender meldet sich einmalig am System an. Single-Sign-On sorgt dafür, dass bei allen folgenden Anmeldevorgängen, z. B. beim Mailprogramm, bei der Warenwirtschaft usw., kein weiteres Eingreifen des Anwenders erforderlich ist.

NoMachine eine Lösung implementiert, an der sich heute alle Nutzer der Software erfreuen können.

3 Fazit: Linux überall?

Linux wird bei der NA nicht als Allheilmittel verstanden. Es gilt das Primat der Sinnhaftigkeit. So gibt es zurzeit keine Pläne, mit *Linux* am Desktoparbeitsplatz zu arbeiten. Dennoch ist *Linux* die Heimat für viele Applikations- und Datenbankserver geworden, die vorher noch im Windows-Umfeld ihr Zuhause fanden. Die Liste der Migrationen auf das Betriebssystem *Linux* hat sich inzwischen auf nahezu 30 Server erhöht.

Es ist nicht zu erwarten, dass die eingeführten, auf *Linux* und OSS basierenden Lösungen in absehbarer Zeit durch gänzlich andere Systeme ersetzt werden. Vielmehr werden aktuelle Entwicklungen wie die Stabilisierung freier Servervirtualisierungslösungen deren Einzug auch in die IT der NA erlauben. Eine gute Grundlage hierfür stellt die geplante Implementierung eines über zwei Rechenzentren verteilten *Storage Area Networks (SAN)* dar.

Gegenstand heutiger Planungen ist bereits die Umstellung auf *Samba 4*. Die Fortschritte bei der Entwicklung werden beobachtet, *Technology Previews*[14] werden regelmäßig evaluiert. Der zurzeit vorhergesagte Fertigstellungstermin von *Samba 4*, das Jahr 2009, erscheint im Sinne dieses aktiven Wartens als in schmerzhaft weiter Ferne gelegen.

Der ausführende Dienstleister, die *inmedias.it GmbH*, ist zwischenzeitlich in die Riege der *RedHat Advanced Business Partner* aufgestiegen. Sämtliche Linux-Server der NA basieren auf verschiedenen RedHat-Produkten. In Zukunft wird das Administratorenteam zusätzlich verstärkt mit dem Know-how des *lsp - linux solution parks*; in diesem Zusammenschluss von vier Hamburger Linux-Dienstleistern wirken derzeit dreißig Linux-Spezialisten.

Literatur

Meyer, J. und Brunke, C. (2005), Migration auf Samba/OpenLDAP bei der Norddeutschen Affinerie AG, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 51–60. http://www.opensourcejahrbuch.de/download/jb2005/chapter_01/osjb2005-01-05-meyerbrunke [09. Jan 2007].

14 Das Samba-Team veröffentlicht regelmäßig so genannte *Technology Previews (TP)*. Zum Zeitpunkt der Entstehung dieses Artikels ist das *TP* Nr. 2 vom 23. März 2006 aktuell. Bis zur Einsatzreife können aber durchaus noch mehrere Jahre vergehen – die Entwickler selbst warnen mit folgenden Worten vor dem produktiven Einsatz: „Samba4 TP is currently a pre-alpha technology. It may eat your cat, but is far more likely to choose to munch on your password database." (vergleiche http://us2.samba.org/samba/history/samba-4.0.0tp3.html)

Kapitel 4

Faszination Technik

„Eine Gesellschaft, die sich auf eine Technik einlässt, braucht eine starke innere Kraft, um von den Zielen nicht verführt, nicht zu gierig zu werden."

– Joseph Weizenbaum

Faszination Technik: Einleitung

NADIM SARROUH

(CC-Lizenz siehe Seite 563)

Seit jeher steht die Technikentwicklung als spezielle Disziplin der Naturwissenschaften vor der Hürde, ihre Produkte gewinnorientiert vermarkten zu können. Das Dilemma ist bekannt: Sobald eine geistige Schöpfung erfolgt ist, entfaltet sie ihren größten Nutzen, wenn ihre Verbreitung nicht eingeschränkt wird. Andererseits sprechen eine Reihe von mehr oder weniger plausiblen Gründen für die Gewährung und Durchsetzung von Eigentumsrechten an geistigen Schöpfungen. Spätestens seit dem späten 19. Jahrhundert sind Patentrechte und andere Schutzrechte an kommerziell verwertbaren geistigen Schöpfungen zur gesellschaftlichen Norm geworden und werden häufig bemüht, wenn es gilt, Rechte an solchen Schöpfungen zu sichern.

Open Source geht einen anderen Weg. Es basiert auf der Offenlegung und der expliziten Aufforderung an die Öffentlichkeit, die Entwicklungen weiterzubenutzen und zu entwickeln. Allerdings zeigt sich, dass Profitinteressen und Vermarktung dadurch nicht in den Hintergrund treten müssen. *IBM*, *Novell* und *RedHat* sind nur die bekanntesten Beispiele dafür, dass sich mit Open Source erfolgreiche Geschäftsmodelle realisieren lassen.

Dieses Kapitel gibt anhand von ausgewählten Beispielen einen Einblick in die Praxistauglichkeit von Open-Source-Lösungen. Im Fokus liegt dabei der technische Bereich, wobei sich „Technik" nicht lediglich auf die Programmiertechniken bezieht, sondern selbst eine breitgefächerte Disziplin darstellt.

Dennis Zimmer, Autor des Buches „VMware und Microsoft Virtual Server" und Senior Consultant bei *Mightycare Solutions*, berichtet mit seinem Kollegen Andreas Niederländer im ersten Artikel dieses Kapitels über die beiden Open-Source-Virtualisierungslösungen, *Xen* und *OpenVZ*. Damit betrachten sie die beiden zurzeit wichtigsten Open-Source-Alternativen zu *VMwares* Virtualierungsumgebungen und erläutern den praktischen Einsatz beider Werkzeuge anhand von Beispielen.

Auch Nico Gulden beschäftigt sich mit dem Trendthema Virtualisierung, aber von der Serverseite aus. Er berichtet von dem LiSoG-Projekt eines „Open-Source–Thin-Clients" und zeigt, dass man mit Open Source auch in der Verwaltung viel erreichen kann. Als Kollege des IT-Leiters der Stadt Schwäbisch Hall, Horst Bräuner,

der im Open Source Jahrbuch 2005 die Migration auf Open-Source-Software in der Stadtverwaltung beschrieb, kann sich Gulden aus erster Hand über die Effektivität und Praxistauglichkeit von Open Source äußern. Er kommt zu dem Ergebnis, dass sich mit Open Source technisch alle Anforderungen realisieren ließen, die zuvor von proprietärer Software erfüllt wurden. Nur: Sind diese Open-Source-Lösungen auch wirklich praktikabel und benutzerfreundlich? Kommen sie über das alte Vorurteil hinweg, Open Source sei nur etwas für „Techies" und „Computer Cracks"? Diese Frage nimmt der nächste Artikel in Augenschein.

Ellen Reitmayr widmet sich in ihrem Beitrag dem Vorwurf, Open-Source-Software sei nicht benutzerfreundlich, versucht die treffenden Punkte der Kritik festzuhalten und anhand des OpenUsability-Projektes Faktoren zu identifizieren, welche die *Usability* von Open-Source-Programmen steigern können. Dabei geht sie nicht sanft mit bisherigen Open-Source-Entwicklern ins Gericht und zeigt, dass ein Gesinnungswandel her muss, um Open Source auch wirklich für den Normalbürger benutzbar zu machen und Vertrauen in Open Source zu schaffen.

Um Vertrauen auf einer anderen Ebene geht es in dem nächsten Artikel. Norbert Pohlmann und Markus Linnemann stellen das Projekt der Forschungs- und Entwicklungsgemeinschaft der *European Multilaterally Secure Computing Base (EMSCB)* namens *Turaya* vor. Auf Open Source basierend handelt es sich bei *Turaya* um eine Sicherheitsplattform, die es gerade aufgrund ihrer offenen Quellen möglich macht, die umstrittene Trusted-Computing-Technologie vertrauenswürdig einzusetzen.

Im nächsten Artikel geht es um das Betriebssystem *NetBSD*. Portabilität steht hier im Vordergrund: Nahezu jeder Rechner lässt sich mit *NetBSD* betreiben und aufgrund der offenen Lizenz wird es häufig in Schulen und Verwaltung eingesetzt. Welche Technik dahintersteckt, beleuchten Huber Feyrer, Stefan Schumacher und Mark Weinem in ihrem Artikel „NetBSD – das portabelste Betriebssystem der Welt".

Einen Abstecher der besonderen Art bietet zu guter Letzt Martin Pichlmair. In seiner Eigenschaft als Medienkünstler und Wissenschaftler gibt er uns einen Einblick in gängige Open-Source-Lösungen im künstlerischen Medienbereich. Vor allem beleuchtet er wichtige Aspekte der Lösungen im Bereich von Gestaltung und Prototypendesign und zeigt die vielfältigen Möglichkeiten auf, die Open Source der Kunst bieten kann.

Es zeigt sich in diesem Kapitel, dass Grundsätze von Freiheit und Offenheit vor allem in der Technik und Softwareentwicklung von großem Vorteil sein können – und das, ohne Profitstreben und Gewinnmaximierung zu verbieten. Weizenbaum warnte im letzten Jahrbuch davor, diese Grundsätze aus den Augen zu verlieren, um nicht in starre Strukturen zu verfallen. Vielleicht kann Open Source dann zu der inneren Kraft der Softwareentwicklung werden, die Joseph Weizenbaum einfordert.

OpenVZ und Xen

Dennis Zimmer und Andreas Niederländer

(CC-Lizenz siehe Seite 563)

Der folgende Artikel behandelt das Thema Virtualisierung im Open-Source–Umfeld. Neben einer Einführung in die Geschichte und Technologie der Virtualisierung werden Architektur und Funktionsumfang der Virtualisierungsprogramme erläutert. Im Kern dieses Artikels stehen die beiden Open–Source-Virtualisierungsumgebungen *OpenVZ* und *Xen*. Für beide Produkte werden Funktionsweise und Charakteristika sowie Installation, Konfiguration und Verwaltung beschrieben. Eine kurze Zusammenfassung der Vor- und Nachteile sowie des Einsatzgebietes soll dem Leser die Gelegenheit geben, das für ihn passende Produkt auswählen zu können. Ein Blick auf andere Virtualisierungsprodukte und die mögliche Entwicklung in der Zukunft runden den Artikel ab.

Schlüsselwörter: Leistungsbewertung · Paravirtualisierung · Virtualisierung · Betriebssystemvirtualisierung

1 Grundlagen der Virtualisierung

Virtualisierung wurde von vielen Analysten als einer der Trends der nächsten Jahre dargestellt, da durch die Virtualisierung immense Vorteile im Serverbetrieb entstehen können. Welche Vorteile rechtfertigen eine solche Präsenz in den Medien und was ist bzw. wie funktioniert Virtualisierung eigentlich? In einem Satz: Virtualisierung stellt die Trennung des Betriebssystems und der darauf betriebenen Applikationen von der physikalischen Hardware dar.

Die Virtualisierungssoftware erstellt eine logische Hardwareumgebung für jede virtuelle Maschine, die komplett unabhängig von der darunterliegenden physikalischen Hardware ist. Daher sind die virtuellen Maschinen komplett von der physikalischen Hardware abgeschottet und können diese nicht kompromittieren. Der Unterschied zwischen der virtuellen und der physikalischen Umgebung ist für die Prozesse und Programme innerhalb der virtuellen Maschine im Normalfall nicht sichtbar.

Die Ansteuerung der Hardware, das Zuweisen der Ressourcen und die Rechteverwaltung regelt die Virtualisierungssoftware. Es besteht keine Möglichkeit für die virtuelle Maschine, aus dieser virtuellen Umgebung auszubrechen. Genau diese Trennung ist auch notwendig, um sichere und effiziente Umgebungen aufzubauen.

2 Virtualisierungstechnologien

Virtualisierung ist zuallererst von *Emulation* zu unterscheiden. Der große Unterschied zwischen Virtualisierung und Emulation liegt darin, dass Emulatoren die gesamte Hardwarearchitektur nachbilden. Dadurch ist es beispielsweise möglich, auf einem x86-System RISC-Prozessoren oder Motorola-CPUs nachzustellen. Dies erfordert jedoch enorme Rechenleistung seitens des Wirtssystems, da jeder Prozessorbefehl sowie die Hauptspeicheradressierung umgesetzt werden müssen.

Virtualisierung verfolgt einen anderen Ansatz: Hier wird nicht die komplette Hardware nachgestellt, sondern nur einzelne Hardwarekomponenten. CPU- und Hauptspeichernutzung können durch die Virtualisierungsprodukte zwar eingeschränkt und partitioniert werden, es erfolgt aber keine Architekturumsetzung. Dadurch wird Performance gewonnen, allerdings wird die Heterogenität eingeschränkt. Auch Virtualisierungstechniken werden in verschiedene Arten aufgeteilt: Hardwarevirtualisierung oder Komplettvirtualisierung, Paravirtualisierung und Betriebssystemvirtualisierung.

Hardwarevirtualisierung Diese Form der Virtualisierung wird von bekannten Unternehmen wie *Microsoft* und *VMware* in ihren Produkten verwendet.[1] Technisch wird bei einer Komplettvirtualisierung jedem Gastsystem eine angepasste Hardware präsentiert. Diese Virtualisierungsschicht präsentiert dem Gastsystem eine immer gleiche Peripherie. Lediglich Prozessor- und Hauptspeicherzugriffe werden direkt an die physikalische Hardware durchgereicht, alle anderen Geräte werden „emuliert". Da diese Emulierung Leistung kostet, werden im Gastsystem entsprechende Hardwaretreiber und Tools installiert, um Leistungsverluste zu minimieren. Dieser Leistungsverbrauch, auch *virtualization overhead* genannt, schmälert die reale Leistung, die den Gastsystemen durch das Wirtssystem zur Verfügung gestellt werden kann.

Paravirtualisierung Der bekannteste Vertreter der Paravirtualisierung ist Xen. Paravirtualisierung unterscheidet sich von der Komplett- oder Hardwarevirtualisierung in dem Sinne, dass hierfür eine Anpassung des Gastsystems notwendig wird. Während die Hardwarevirtualisierung der virtuellen Maschine eine komplette Umgebung mit Standardgeräten zur Verfügung stellt, merkt das Gastbetriebssystem nichts von der Virtualisierung und muss nicht angepasst werden. Paravirtualisierung basiert jedoch darauf, dass die virtuelle Maschine mit der speziell

1 Spezielleres zu *Microsoft Virtual Server* und VMware-Produkten ist in Zimmer (2005) zu finden.

für die Virtualisierung kreierten virtuellen Hardware kommunziert. Diese wird durch die Virtualisierungsschicht (*VMM* oder *Virtual Machine Monitor*) erstellt. Dies kann sie jedoch nur, wenn der Betriebssystemkern (*Kernel*)[2] so angepasst wird, dass dieser statt mit der physikalischen Standardhardware mit der virtualisierten Hardware kommuniziert. Dadurch muss die physikalische Hardware nicht für jede einzelne virtuelle Maschine virtualisiert werden, sondern die Gastbetriebssysteme greifen direkt auf eine angepasste Hardware zu. Hierbei nutzen *Hypervisor*[3] und Gastsysteme immer die gleichen Treiber.

Betriebssystemvirtualisierung Die Betriebssystemvirtualisierung geht einen anderen Weg und virtualisiert nicht ein Hardwaresystem, sondern partitioniert das Betriebssystem in mehrere, von der Basis her gleiche Teile. Vorteil der Betriebssystemvirtualisierung ist, dass idealerweise bei der Erstellung der Gastsysteme nur die Individualdaten der virtuellen Maschine angelegt werden. Alle „gleichen" Betriebssystemdaten (Programmbibliotheken etc.) des Wirtssystems werden durch die Gäste mitgenutzt. Dies gilt ebenfalls für die laufenden Prozesse. Weitere Anpassungen des Gastbetriebssystems sind in keiner Form nötig. Abweichende Daten werden im Heimatverzeichnis der virtuellen Maschine abgelegt. Mittels dieser Technik besteht eine virtuelle Maschine in der Grundausstattung nur aus sehr wenigen Dateien und verbraucht deutlich weniger Hauptspeicher als ein normal installierter Server. Betriebssystemvirtualisierung steht für eine sehr hohe Ausnutzung der Wirtssystemressourcen, d. h. einen geringen Virtualisierungsoverhead (1–3 %). Darüber hinaus ist aufgrund der geringen Leistungsansprüche der Gäste eine sehr hohe Anzahl virtueller Maschinen auf verhältnismäßig schwacher Hardware möglich (ca. Faktor zehn gegenüber Komplettvirtualisierung). Größter Nachteil ist die Abhängigkeit des Gastsystems vom Wirtssystem, da eine homogene Basis vorausgesetzt wird. Wichtigster Aspekt auch dieser Form der Virtualisierung ist, dass alle Gäste voneinander getrennt und unabhängig agieren können.

3 OpenVZ

OpenVZ[4] gehört zu den Betriebssystemvirtualisierungsprodukten, partitioniert also das Wirtsbetriebssystem und nutzt dieses auch in Teilen mit (siehe Abbildung 1). Entstanden aus dem kommerziellen Virtualisierungsprodukt *Virtuozzo* von *SWsoft*, ermöglicht OpenVZ die Virtualisierung von *Linux*. *SWsoft* hat aus OpenVZ ein Community-Projekt gemacht, welches auf ein stetig wachsendes Interesse stößt. OpenVZ

2 Der *Kernel* ist das Kernstück des Betriebssystems, siehe dazu http://de.wikipedia.org/wiki/Betriebssystemkern.

3 *Hypervisor* ist ein Synonym für *VMM*.

4 http://www.openvz.org

ist quelloffen, kostenfrei und steht unter der *GPL* bzw. *QPL*[5]. Ziel dieses Projektes ist es, Bestandteil des offiziellen Linux-Kernels zu werden. Einen Teilerfolg hat OpenVZ schon erzielt, da OpenVZ zukünftig in der Distribution *Debian*[6] enthalten sein wird.

3.1 OpenVZ-Verwaltung

Die Verwaltung der Virtualisierung erfolgt ausschließlich mithilfe mächtiger Kommandozeilentools, die auch durch Skripte automatisiert werden können. Darunter finden sich neben den Basistools zur Verwaltung auch Zusatzprogramme zur Ressourcenkontrolle oder der Templateverwaltung. Nachteil von OpenVZ ist dabei die zwingende Homogenität: Auf einem Kernel-2.6-Wirtsbetriebssystem können nur Gastsysteme, die auf dem gleichen Kernel basieren, betrieben werden. Das *Virtual Environment (VE)*[7] wird mittels eines Templates erstellt, das eine Kernkomponente von OpenVZ darstellt. In einem solchen Template sind alle Dateien des Linux-Systems eingebunden, die benötigt werden, um die *VE* auf dem Wirtssystem zu betreiben. Es wird immer nur der Kernel des laufenden Wirtssystems genutzt. Grundsätzlich existiert immer nur ein Template pro OpenVZ-Server. Alle darauf basierenden *VEs* nutzen diesen und legen Änderungen gegenüber diesem System in einem eigenen Verzeichnis ab. Unter diesen Änderungen sind Softwareinstallationen und Konfigurationsänderungen zu verstehen, z. B. die Anpassung der Konfigurationsdatei „/etc/fstab“. Dadurch wird eine höhere Speicherdichte (ca. 100 MB pro *VE* ohne Applikationen) erreicht, durch die viel Speicherplatz eingespart werden kann.

Dieses Verfahren ist beim „großen Bruder“ *Virtuozzo* nochmals deutlich verbessert worden, resultierend in einem Festplattenbedarf der *VEs* im Bereich von 10–30 Mbyte (Standardinstallation ohne Applikationen). Die Templates werden für die gängigsten Linux-Derivate zum Download auf der Webseite bereitgestellt. Weiterhin ist es bei entsprechender Kenntnis auch möglich, Templates mittels des Programms *vzpkg* selbst zu erstellen. Die Vermutung, es würden Sicherheitslücken durch die Nutzung von gleichen Daten (Template) und Prozessen entstehen, bestätigt sich nicht. Derzeit ist keine Möglichkeit bekannt, aus dem Gastsystem in andere Gäste oder gar den Wirt zu gelangen. Es herrscht eine vollkommene Trennung der Systeme. Allerdings sollte das Wirtssystem wie bei jeder Virtualisierung sehr gut abgeschottet und ausfallsicher aufgestellt werden. Dies gilt sowohl für die Hard- als auch für die Software. Zusätzlich ist es möglich, die maximal nutzbaren Ressourcen des Wirtssystems pro virtueller Maschine einzuschränken, was in vielen Fällen sehr nützlich sein kann, wenn z. B. *VEs*

5 *Q Public License* – eine von der Open-Source-Initiative zertifizierte Softwarelizenz, die sich von der bekannten *GNU General Public License (GPL)* unterscheidet. Man darf Software, die unter der QPL veröffentlicht wurde, in unveränderter Form kopieren und weitergeben, solange man das Paket unangetastet lässt – d. h. Copyright-Hinweise, Warenzeichen und Haftungsauschlüsse müssen mitgegeben werden, so wie sie der Hersteller angegeben hat. Dies sichert die Integrität der Originalversion. Siehe auch http://de.wikipedia.org/wiki/Q_Public_License.

6 http://www.debian.org

7 *VE* bezeichnet die virtuelle Maschine unter OpenVZ.

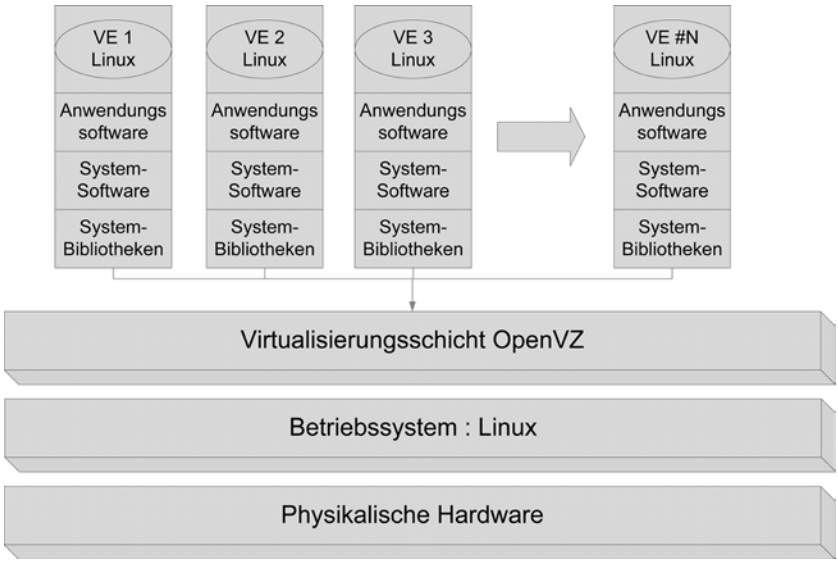


Abbildung 1: Die OpenVZ-Architektur

Dritten zur Verfügung gestellt werden. Verschiedene Peripherieformen sind aufgrund dieser Technik jedoch derzeit eingeschränkt, wodurch der Einsatz von beispielsweise *USB* oder paralleler Schnittstelle im Gast nicht möglich ist. Seit Juni 2006 wurde insbesondere die Netzwerk-Unterstützung weiterentwickelt, so dass OpenVZ nun *IPv6* und *Bridging* unterstützt und somit eine breitere Anwendungsunterstützung ermöglicht.

3.2 Einsatzgebiete von OpenVZ

OpenVZ ist aufgrund der Homogenität beim Einsatz auf reine Linux-Umgebungen beschränkt. Für die Integration in eine bestehende Linux-Installation existieren auf der offiziellen Webseite und auch auf verschiedenen Community-Webseiten detaillierte Anleitungen. Ohne diese sind fortgeschrittene Linux-Kenntnisse zwingend notwendig, da der zu startende Kernel ausgetauscht werden muss. Ist das System erst einmal installiert, präsentiert sich OpenVZ sehr zuverlässig und performant. Es können durchaus auch produktive Systemumgebungen virtualisiert werden. Dennoch ist der Haupteinsatzbereich in kleineren Umgebungen, insbesondere Test- und Entwicklungsumgebungen, zu sehen.

Mit der neuesten Version wurden zwei sehr interessante und wichtige Funktionalitäten eingeführt: virtuelle Ethernet-Netzwerke und Live-Migration. Vormals bestand nur die Möglichkeit eines virtuellen Netzwerks, das nur eine logische Trennung durch

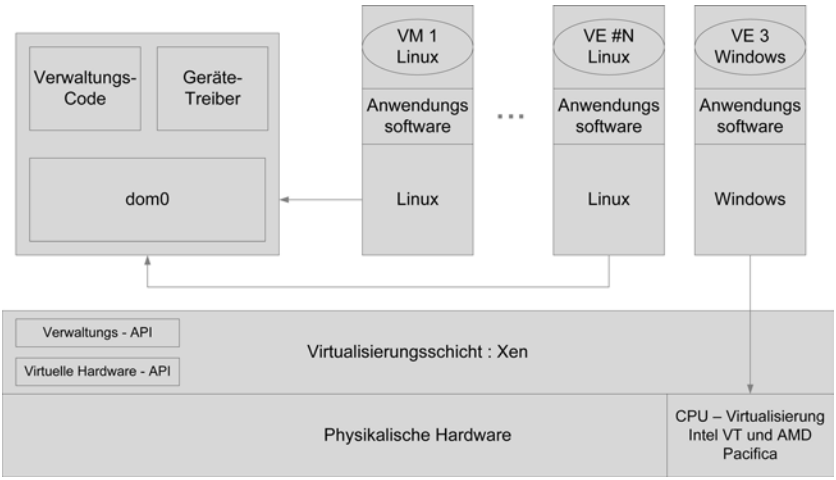


Abbildung 2: Die Xen-Architektur

IP-Subnetze zuließ. Mit den nun verfügbaren virtuellen Netzwerken kann die virtuelle Maschine deutlich flexibler genutzt werden, z. B. durch die Verwendung eigener MAC-Adressen und Netzwerknamen. Live-Migration bedeutet das Verschieben eines *VE* zwischen zwei OpenVZ-Servern ohne ein Herunterfahren der virtuellen Maschine. Jedoch muss diese derzeit noch kurz angehalten werden, was zu einem minimalen Ausfall führt. Allerdings ist derzeit der Einsatz von Zusatzkarten oder Peripherie (z. B. Faxkarten) meist nicht möglich.

4 Xen

Xen[8] verfolgt gegenüber OpenVZ einen anderen Weg: Ähnlich wie bei den Produkten aus dem Hause *VMware* wird hier für jeden Gast ein eigener Kernel gestartet. Das bedeutet prinzipiell, dass auf einem Wirt verschiedene Betriebssysteme gleichzeitig gestartet werden können. Dies ist allerdings auf normaler x86-Hardware ohne Anpassung des Gast-Kernels nicht möglich. Schuld daran sind die sogenannte „Ring-0-Befehle" der x86-Prozessoren, die exklusiv dem Kernel eines Betriebssystems vorbehalten sind. Um mehrere Betriebssysteme zu starten, ist es daher unumgänglich, diese Befehle aus dem Kernel des Gastes zu entfernen bzw. zu ersetzen. Während *VMware* dies *on-the–fly* beim Betrieb des Gastes macht, muss dieser bei Xen vor dem Start entsprechend angepasst werden („Paravirtualisierung").[9]

8 http://www.xensource.com

9 Eine detaillierte Beschreibung der Funktionsweise von Xen und der Einrichtung findet sich in Radonic und Meyer (2006).

Xen selbst besteht eigentlich nur aus einer „dünnen“ Virtualisierungsschicht, dem *VMM (Virtual Machine Monitor)*, der einzeln nicht zu nutzen ist (siehe Abbildung 2). Die virtuellen Systeme werden bei Xen jeweils als *Domain* bezeichnet. Nach dem Start des *VMM* benötigt Xen einen besonderen Gast, der direkten Zugriff auf die I/O-Bereiche der Hardware erhält. Dieser Gast wird als *Dom0* bezeichnet und dient auch der Kontrolle der anderen Gäste. Alle anderen Gäste – auch *DomUs* genannt – bedienen sich der Hardware, die durch die *Dom0* zur Verfügung gestellt wird. Somit kann die *Dom0* von der Funktionsweise her auch als Wirt-System bezeichnet werden.

Auf Xen werden große Hoffnungen gesetzt, da viele namhafte Hardware- und Betriebssystemhersteller die Unterstützung dieses Produktes zugesichert haben, darunter *Intel*, *IBM*, *RedHat* und *Microsoft*. Für einen Einsatz in Unternehmen sind Herstellersupport sowie eine zentrale Verwaltungsoberfläche zwingend erforderlich. Beides wird durch das Unternehmen *XenSource* angeboten, jedoch fehlen zur kürzlich erschienenen *XenEnterprise* noch Erfahrungswerte. Um Xen einzusetzen, muss also zurzeit auf komfortable Management-Software verzichtet werden. Aber auch mithilfe der mitgelieferten Befehlszeilentools und der rudimentären Konfiguration, die z. B. *SUSE Linux* mitbringt, ist es durchaus möglich, eine leistungsfähige Xen-Umgebung umzusetzen.

Red Hat plant grundsätzlich, Xen in sein Enterprise-Produkt *RHEL* zu integrieren, verzögert die Auslieferung jedoch noch. Nach Aussagen von *Red Hat* spielen hier die noch fehlende Stabilität und Lücken im Sicherheitsbereich eine Rolle. Geplant ist die Integration von Xen in *RedHat Enterprise Linux 5*.

4.1 Einsatzgebiet von Xen

Aktuell wird Xen hauptsächlich für die Virtualisierung von Linux-Systemen eingesetzt, da mit *Linux* aufgrund der offenen Quellen Paravirtualisierung möglich ist und Distributionen Werkzeuge zur Installation und Administration mitbringen. *SUSE Linux Enterprise Server 10* kann quasi *out of the box* als Xen-Wirt inkl. virtueller Linux–Maschinen über eine grafische Oberfläche installiert werden. Das macht den Einstieg einfacher. Die Einsatzgebiete für solche Maschinen sind überall dort zu finden, wo aktuell Linux-Maschinen ihren Dienst tun. Einfache Maschinen ohne besonderen Ressourcen-Bedarf (z. B. DNS- oder DHCP-Server) können bereits jetzt ohne Bedenken in einer Xen-Umgebung eingesetzt werden.

Ein typisches Szenario für System-Administratoren ist das Update von älteren Linux-Systemen auf eine neue Distribution. Oft sind alte Maschinen mit vielen Diensten ausgestattet, die dabei für den Betrieb des Netzes laufen müssen. Für ein Update auf eine neue Maschine sind ausgiebige Tests, möglichst kurze Ausfallzeiten bei der Umstellung sowie ein Rollback-Szenario nötig. Etwas entspannter ist die Situation mit Hilfe von Virtualisierung.

Zunächst wird ein neues Linux-System installiert und das alte System in einer virtuellen Umgebung gestartet. Im Idealfall ändert sich dadurch zunächst nichts für

die Clients im Netzwerk: Sie greifen immer noch auf den alten Server zu – mit dem Unterschied, dass der Server virtuell ist. Im zweiten Schritt installiert man – neben dem alten Server als Gast – weitere virtuelle Maschinen, auf die man dann die Dienste des alten Servers nach und nach migriert, bis der alte Server dann ganz deaktiviert werden kann. Ein Rollback auf die alte Konfiguration ist hier dann ganz einfach: Man aktiviert den entsprechenden Dienst einfach wieder auf der alten (virtuellen) Maschine.

Auf die Problematik des Zugriffs mehrerer Betriebssysteme auf den Prozessor und der damit notwendigen Anpassung des Gastbetriebssystemkernels haben die beiden großen Hersteller von x86-kompatiblen Prozessoren reagiert. So gibt es seit kurzem Prozessoren, die es erlauben, ein Gastbetriebssystem ohne Anpassung des Kernels zu betreiben. *Intel* nennt diese Technik „Vanderpool Technology (VT)" und *AMD* „Pacifica". Aktuelle Prozessoren der beiden Hersteller sind bereits am Markt verfügbar. Die großen Serverhersteller beginnen seit Sommer 2006, ihre Produkte mit *VT* bzw. *Pacifica* auszustatten. Mit dieser Technik, die auch „vollständige Virtualisierung" genannt wird, ist es prinzipiell möglich, jedes x86-Betriebssystem in einer virtuellen Umgebung zu starten – also auch *Windows*.

4.2 Vor- und Nachteile von Xen

Der Einsatz von Virtualisierung mit Xen hat auch seine Nachteile. Das wohl größte Manko ist derzeit die noch fehlende Integration in die Wirtssysteme. Ansprechende Konfigurations- und Administrationsoberflächen, wie man sie beispielsweise von *VMware*[10] gewohnt ist, sucht man vergeblich. Auch die Möglichkeit, ein oder mehrere *Snapshots* eines Gastes zu erstellen, bietet Xen von sich aus nicht. Hierzu muss man auf entsprechende Techniken des darunterliegenden Betriebssystems des Wirts – z. B. *LVM Snapshots* – zurückgreifen. Ein weiterer Nachteil ist die feste Partitionierung des Speichers. Es ist nicht möglich, dass die Gäste den zur Verfügung stehenden Hauptspeicher dynamisch nutzen können. Vielmehr gibt es hier nur eine starre Partitionierung, die für einen Gast immer denselben Anteil an Speicher belegt, egal, ob der Gast diesen Speicher nun braucht oder nicht.

Trotzdem kann Xen in vielen Bereichen Vorteile aufweisen. Durch die dünne Virtualisierungsschicht arbeiten die Gäste nahe an der Hardware, ein Aspekt, der sich vor allem in der Gesamtperformance der Gäste niederschlägt. Benchmarks mit unserer Testhardware haben ergeben, dass der Zugriff auf *Block-Devices* – hier unsere SATA-Platten – nur etwa 10 Prozent langsamer ist als im Wirtssystem. Zugriffe auf das RAM des Gastes waren in unseren Tests genauso schnell wie auf dem Wirt. Kein Wunder, da der Wirt (als *Dom0*) ja letztendlich auch nur ein Gast des *VMM* ist. Die „Workload-Protection" genannte Technik erlaubt es dem Administrator sogar, die Zugriffe auf die Ressourcen mit entsprechenden Prioritäten zu versehen und Grenzen für den Ressourcenverbrauch (insbesondere CPU) festzulegen. Somit sollen z. B.

10 Mehr zu dem *VMware Server* und dem *VMware Player* in Zimmer (2006)

Denial-of-Service-Angriffe auf einen Gast nur begrenzten Einfluss auf andere Gäste auf der gleichen Maschine haben. Weitere Pluspunkte im Bereich Lastverteilung kann Xen durch die bereits erwähnte Live-Migration sammeln. Die breite Unterstützung durch namhafte Soft- und Hardwarehersteller unterstreicht das hohe Potenzial von Xen im professionellen Einsatz.

4.3 Zukunftsperspektiven

Es ist davon auszugehen, dass Xen das Open-Source-Virtualisierungsprodukt der nächsten Jahre wird. Es sieht so aus, als könnten sich die Konkurrenten wie *AMD* und *Intel* oder auch *Microsoft* und die Open-Source-Community auf diesen neutralen „Standard" einigen. Wir können also davon ausgehen, dass die Integration von Xen fester Bestandteil zukünftiger Linux-Distributionen sein wird und dass es in Zukunft nur noch Server mit der notwendigen CPU-Unterstützung geben wird. Dadurch wird sich auch das Angebot an kommerziellem Support erweitern, insbesondere für die Enterprise-Distributionen. Ob Xen allerdings langfristig bezüglich Umfang, Stabilität und Zertifizierung mit *ESX*, dem weit verbreiteten und äußerst leistungsfähigen Virtualisierungsprodukt von *VMware*, vergleichbar sein wird, bleibt offen.

5 Fazit

Virtualisierung ist und bleibt ein zukunftsträchtiges Thema. Die beiden Produkte OpenVZ und Xen können sowohl bei Einzelanwendern als auch in professionellen Infrastrukturen bei entsprechender Planung eingesetzt werden. Aufgrund der Weiterentwicklung durch die Community und professionelle Unternehmen werden die Produkte stetig verbessert und nehmen an Funktionsumfang zu. Vor allem Xen wird in der nächsten Zeit in vielen Linux-Distributionen zu finden sein. Einige Virtualisierungshersteller nutzen schon jetzt die Xen-Basis, um kommerzielle Produkte und Leistungen anzubieten.

Literatur

Radonic, A. und Meyer, F. (2006), *XEN3*, 1. Aufl., Franzis' Verlag GmbH, Poing.

Zimmer, D. (2005), *VMware und Microsoft Virtual Server*, 1. Aufl., Galileo Press, Bonn.

Zimmer, D. (2006), *VMware Server und VMware Player*, 1. Aufl., Galileo Press, Bonn.

Prototyp eines Open-Source-basierten Thin-Clients*

NICO GULDEN

(CC-Lizenz siehe Seite 563)

Eine der Wachstumsbranchen bleibt auch in Zukunft der Thin-Client-Markt.[1] Die Vorteile von Thin-Clients[2] liegen auf der Hand: Zentrales Anwendungssoftware-Management, längere Nutzungsdauer, höhere Zuverlässigkeit und ein günstigerer Preis im Vergleich zu PCs sprechen für den Einsatz von Thin–Clients und *server-based computing*. Fokus des LiSoG-Szenarios „Thin-Client" sind Linux-basierte Thin-Clients. Im Linux- und Open-Source-Bereich sieht die *LiSoG* einen Bedarf an dieser Form von Computerarbeitsplätzen und widmet sich der Beschreibung eines Linux-basierten Open-Source-Thin–Clients, um Machbarkeit und Handlungsbedarf aufzuzeigen.
Der Thin-Client-Prototyp der Stadt Schwäbisch Hall zeigt, dass die Bestandteile für einen Linux-basierten Open-Source-Thin-Client vorhanden sind. Anschließend werden noch die bestehenden Probleme, wie z. B. eine zentrale Systems-Management-Software, angesprochen.

Schlüsselwörter: Linux Terminal Server · server-based computing · FreeNX

1 Einleitung

Ein aktueller Trend in der Informationstechnologie ist die Virtualisierung auf allen Ebenen. Ein Teil davon ist die Rückbesinnung zur Zentralisierung der Administration von Computern durch *server-based computing*[3]. Dadurch werden in vielen Abteilungen von Firmen, Kommunen und Organisationen voll ausgestattete Computer,

* Dieser Artikel basiert auf dem Arbeitspapier „LiSoG-Szenario Thin Client, Prototyp eines Open Source basierten Thin Clients" des *Linux Solution Group e. V.*, erhältlich unter http://www.lisog.org/publikationen/lisog-thin-client.pdf.

1 Vergleiche auch http://www.etcf.de/cms/presse_meldungen_mitglieder_2005_de/detail.php?nr=5058&kategorie=presse_meldungen_mitglieder_2005_de.

2 Ein Computer als Endgerät (Terminal), dessen funktionale Ausstattung auf die Ein- und Ausgabe beschränkt ist.

3 Die Ausführung der Anwendungen erfolgt nicht mehr auf dem Client, sondern auf dem Server.

so genannte „Fat-Clients“, unnötig. Es kommen zunehmend Pläne auf, PC-basierte Rechnerumgebungen durch Thin-Clients zu ersetzen. So strebt beispielsweise die Stadt Schwäbisch Hall eine Umstellung ihrer Desktop-PCs zu Thin-Clients in Teilen ihrer IT-Infrastruktur an (Bräuner 2005, S. 48). Auch die Stadt Freiburg erwägt im Zuge der eigenen Hardwarestandardisierungsstrategie den Einsatz von Thin-Clients als einfache Systeme für *server-based computing*.

Da im Störungsfall die Endgeräte ohne viel Aufwand ausgetauscht werden können, wird bei der Administration viel Zeit eingespart. Die Anwendungskonfiguration erfolgt bei *server-based computing* nur noch zentral auf dem Terminalserver. Die Konfiguration auf den Client-Endgeräten entfällt. Hinzu kommen der niedrige Stromverbrauch, die geringe Wärmeentwicklung und die daraus resultierende geringe Geräuschentwicklung.

Das LiSoG-Szenario „Thin-Client“ beschreibt einen komplett Linux-basierten Open-Source-Thin-Client. Es befasst sich vorrangig mit einem aus Standard-Komponenten bestehenden Software-Stack[4] auf der Thin-Client-Hardware. Dieses Softwarepaket soll leicht portierbar und für die Massenproduktion geeignet sein.

Der Prototyp der Stadt Schwäbisch Hall des Open-Source-basierten Thin-Clients bietet einen aktuellen Linux-Kernel zusammen mit aktuellen Softwarepaketen. Damit wird ein einheitliches Hardwarehersteller-unabhängiges System geschaffen, das leichter in bestehende Administrationsinfrastrukturen integrierbar ist.

Grundlage hierfür ist ein Linux-Betriebssystem in Minimalausführung, das ausschließlich den Betrieb der Thin-Client-Hardware und die Kommunikation zum Server unterstützt. Mit dem Prototyp der Stadt Schwäbisch Hall ist eine effiziente Thin–Client-Umgebung implementiert worden. Sie wird in diesem Dokument im Detail beschrieben. Darauf aufbauend werden Zukunftstechnologien diskutiert.

2 Prototyp der Stadt Schwäbisch Hall

Ein durchschnittlicher Computerarbeitsplatz in einem Unternehmen stellt dem Mitarbeiter einen PC zur Verfügung. Dieser PC verfügt über die notwendige Software, damit der Mitarbeiter seine tägliche Arbeit erledigen kann. Dazu zählen mindestens ein E-Mail-Programm, ein Webbrowser, eine Office-Anwendung und andere Software. Die Daten, wie beispielsweise Office-Dokumente, liegen hierbei meist zentral auf einem Server. Dies gilt prinzipiell für Windows- und Linux-Arbeitsplätze.

Architektonisch gesehen verfügt der Linux-Arbeitsplatz über eine Besonderheit. Für die grafische Darstellung von Anwendungen, wie Webbrowser und E-Mail-Programm, kommt das *X Window System* zum Einsatz.

4 Unter einem Software-Stack versteht man eine Zusammenstellung aufeinander aufbauender Softwareprodukte. Ein bekanntes Beispiel ist *Linux/Apache/MySQL/PHP (LAMP)*.

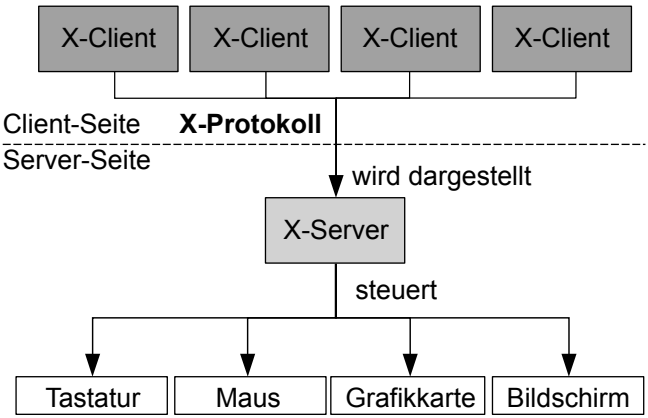


Abbildung 1: X Window System

2.1 X Window System

Das *X Window System (X11)* ist ein netzwerkfähiges Fenstersystem. Es wird im Allgemeinen unter dem Betriebssystem *UNIX* eingesetzt. Entwickelt wurde es am *MIT* (*Massachusetts Institute of Technology* in Cambridge). Für detaillierte Informationen zum *X Window System* sei auf *X.Org*[5], den Nachfolger von *XFree86*, verwiesen. Das Projekt *Freedesktop.org*[6] beschäftigt sich ebenfalls mit vielen Aspekten des *X Window System*. Um die Darstellung des Webbrowser-Fensters kümmert sich dieses *X Window System*. Es basiert auf dem Client-Server-Modell, bestehend aus X-Server und X-Client und ist netzwerktransparent.

Der X-Server kommuniziert mit den Geräten für die Ein- und Ausgabe, zum Beispiel Maus, Tastatur, Grafikkarte und Bildschirm. Über eine rastergrafikbasierte Darstellung erfolgt das Zeichnen von Linien und Mustern. Der X-Server ist zuständig für die ereignisorientierte Handhabung von Maus oder Tastatur. Unter UNIX-Systemen ist der X-Server die Grundvoraussetzung zur Darstellung von grafischen Anwendungen und stellt somit ein Minimalsystem zur Verfügung. *XFree86* und *X.Org* sind Implementierungen des X-Servers.

Jedes grafische Anwendungsprogramm, wie beispielsweise der Webbrowser *Mozilla Firefox*, ist ein X-Client. Der X-Server übernimmt die Darstellung des X-Clients und reicht ihm verschiedene Ereignisse wie etwa Tastatureingaben und Mausbewegungen weiter.

Abbildung 1 zeigt das Client-Server-Modell des *X Window System*. Die Kommunikation über die gestrichelte Linie in der Abbildung läuft netzwerktransparent über

5 http://www.x.org
6 http://www.xserver.freedesktop.org

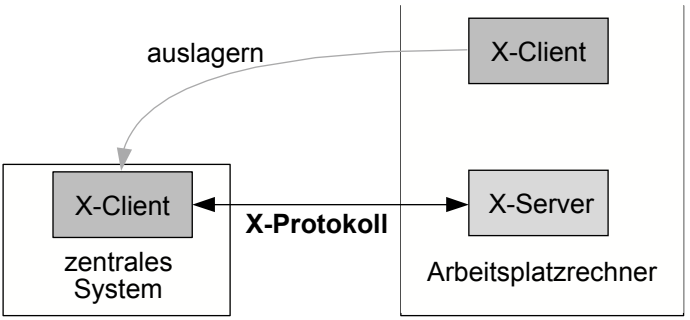


Abbildung 2: Physische Trennung von X-Client und X-Server

das X-Protokoll ab. X-Server und X-Client können auf demselben Computer ausgeführt werden, wie in Abbildung 1. Das ist bei Desktop-Systemen der Fall und die häufigste Anwendung. Angenommen, auf einem PC wird ein *Linux* mit einem KDE- oder GNOME-Desktop installiert, dann befinden sich X-Server und die X-Clients auf einem PC. Die X-Clients sprechen den lokalen X-Server an und dieser übernimmt die Darstellung.

Durch das Client-Server-Modell ist es dem *X Window System* außerdem möglich, beide Komponenten auf unterschiedlichen Systemen zu betreiben. So können die Anwendungen in Form der X-Clients zentral auf ein System ausgelagert und verwaltet werden und deren Darstellung findet dezentral auf dem X-Server des Arbeitsplatzsystems statt, wie Abbildung 2 zeigt.

Durch die Trennung von X-Client und X-Server benötigt der Arbeitsplatzrechner weniger Ressourcen in Form von Arbeitsspeicher und Festplattenplatz. Er wird zu einem „dünnen" Arbeitsplatzrechner, einem *Thin-Client*. Das Thin-Client-Konzept macht von dieser Trennung Gebrauch. Auf dem Thin-Client läuft der X-Server, der die Steuerung der Hardware bestehend aus Maus, Tastatur, Grafikkarte und Bildschirm übernimmt. Auf einem zentralen System, dem Terminalserver, laufen die X-Clients, deren Ausgaben durch das X-Protokoll über das Netzwerk zum X-Server des Thin-Clients weitergeleitet werden und vor dem Benutzer auf dem Bildschirm zur Darstellung kommen. Abhängig von der Geschwindigkeit des Netzwerks ist es möglich, das komplette Betriebssystem inklusive X-Server beim Start des Arbeitsplatzrechners über das Netzwerk zu laden. Das erspart den Festplattenspeicher. Zu Beginn eines Arbeitstages wird jedoch das Netzwerk übermäßig stark belastet, da beim fast zeitgleichen Hochfahren der Arbeitsplatzrechner durch die Mitarbeiter für jeden Rechner, das Betriebssystem über das Netzwerk geladen werden muss. Bei einem schwachen Netzwerk ist es unumgänglich das Betriebssystem mit dem X-Server auf dem Thin-Client zu speichern.

Durch *Multicast TFTP*[7] kann die Last reduziert werden. Dies wird im Novell-Produkt *NLPOS*[8] bereits eingesetzt und zum Beispiel von Kaufhausketten für das morgendliche Inbetriebnehmen von mehreren hundert Kassensystemen über 10-Mbit/s–Netze genutzt. Zusätzlich kann komprimierte Übertragung das Netzwerk entlasten.

Für die physische Trennung von X-Server und X-Client ist ebenso ein leistungsfähiges Netzwerk für das X-Protokoll notwendig, andernfalls wirkt die Darstellung auf dem Thin-Client langsam und zäh. Hierbei ist wiederum zu berücksichtigen, dass mehrere Mitarbeiter gleichzeitig an ihren Thin-Clients arbeiten und in ständigem Kontakt mit dem zentralen System stehen.

2.2 Die NX-Technologie

Die NX-Technologie setzt an dem Bandbreitenhunger des X-Protokolls an und verpasst ihm eine „Diät".[9] Für die Kommunikation zwischen Client und Server über das X-Protokoll ist jedoch ausreichend Bandbreite notwendig, und über Modem- oder ISDN-Verbindung erscheint dem Anwender die Anwendung eher zäh und langsam, was unter anderem durch die vielen Operation-Codes verursacht wird, die das X-Protokoll definiert (Bräuner und Pfeifle 2004).

NX ist ein Aufsatz zum X-Protokoll und steigert die Kommunikationseffizienz durch folgende Maßnahmen:

- Die Übertragungsdaten werden komprimiert.
- Es wird ein Cache für bereits übertragene Daten angelegt.
- Die „Roundtrips" zwischen X-Client und X-Server werden reduziert.

Das X-Protokoll definiert ca. 160 verschiedene Opcodes. Unter Opcodes versteht man Anforderungen (*requests*) des X-Clients an den X-Server (oder umgekehrt, dessen Mitteilungen an den X-Client). Viele X-Client-Anforderungen erwarten eine Antwort (*reply*) vom X-Server. Zusammengehörige Paare von *X Client requests* und *X server replys* bilden jeweils einen „Roundtrip". Allerdings erfordert nicht jeder *request* auch tatsächlich einen *reply*.

Abbildung 3 zeigt die Architektur der NX-Technologie. Eine entfernte X-Anwendung in Form eines X-Clients ist mit einem NX-Server über das X-Protokoll auf dem Server verbunden (rechts im Bild). Die X-Anwendung kann auch eine vollständige Desktop-Sitzung sein, wie beispielsweise *KDE* oder *GNOME*. Der NX-Server puffert und komprimiert die Daten und schickt sie verpackt im NX-Protokoll an den Client (links im Bild). Der NX-Client entpackt und puffert die Daten und gibt sie über das X-Protokoll an den lokalen X-Server weiter.

7 Das *Trivial File Transfer Protocol (TFTP)* ist ein sehr einfaches Dateiübertragungsprotokoll.

8 *Novell Linux Point of Service*: http://www.novell.com/documentation/nlpos9.

9 *NX* ist eine fast vollständig unter der *GNU GPL* stehende Remote-Desktop-Software der italienischen Firma *NoMachine*. Mit *NX* kann man den Bildschirminhalt eines entfernten Computers auf einen lokalen Rechner (auch betriebssystemübergreifend) übertragen und damit arbeiten.

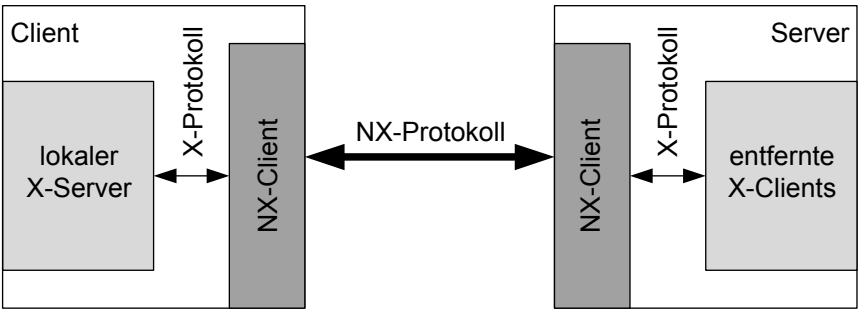


Abbildung 3: Architektur der NX-Technologie

Mit dem Client-Server-Modell von *NX* lassen sich ähnliche Szenarien konstruieren wie mit dem *X Window System*. Abbildung 4 zeigt eine Beispielumgebung. Der NX-Server fungiert als „Proxy" für die Protokolle *RDP*[10] und *VNC*[11] und für das X-Protokoll und kapselt sie in das NX-Protokoll. Auf den Clients ist jeweils ein NX–Client installiert, der die Darstellung vom NX-Server auf den lokalen X-Server bringt. *NX* ermöglicht mit einem Client die Darstellung verschiedener Terminaldienste über dasselbe Protokoll. Hierbei ist es außerdem möglich, das NX-Protokoll verschlüsselt über *SSH*[12] zu betreiben. Da in vielen Firewalls der SSH-Port 22 freigeschaltet ist, entfällt das Öffnen der Firewall für einen weiteren Port.

Die NX-Technologie vermindert den Netzwerkverkehr zwischen einem X-Client und einem X-Server und ermöglicht somit die physische Trennung dieser beiden über durchsatzschwache Leitungen, wie Modem, ISDN oder GPRS. Die Darstellung der Anwendung erscheint nicht mehr zäh, sondern läuft flüssiger ab als nur über das X-Protokoll. Der Anwender erlangt den Eindruck, die Anwendung würde direkt auf dem Arbeitsplatzrechner vor ihm laufen.

Für mehr Informationen zu *NX* sei auf die Firma *NoMachine*[13] verwiesen. Dort finden sich weitere Dokumentation, Beispiele und Screenshots. Bei *BerliOS* findet sich eine Sammlung zu Fakten über *NX* (OpenFacts 2006). Neben dem NX-Client von *NoMachine* existiert noch *FreeNX* auf *Berlios*[14].

10 Das *Remote Desktop Protocol* ist ein Protokoll von *Microsoft*.

11 *Virtual Network Computing* ist eine von *AT&T* entwickelte Software, die den Bildschirminhalt eines weit entfernten Rechners auf einem lokalen Rechner anzeigt und Tastatur- und Mausbewegungen des lokalen Rechners an den entfernten Rechner sendet.

12 *Secure shell* oder *SSH* ist sowohl ein Programm als auch ein Netzwerkprotokoll, mit dessen Hilfe man sich über eine verschlüsselte Netzwerkverbindung auf einem entfernten Computer einloggen und dort Programme ausführen kann. Die *IANA* hat dem Protokoll den TCP-Port 22 zugeordnet.

13 http://www.nomachine.com

14 http://freenx.berlios.de

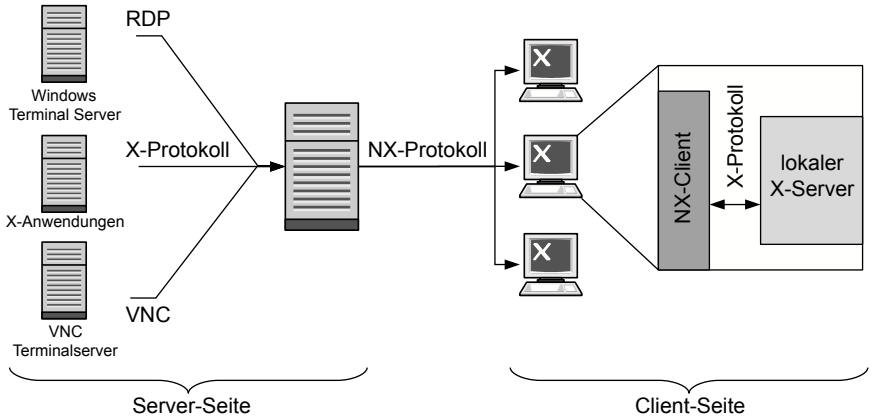


Abbildung 4: NX-Umgebung

2.3 Thin-Client

Für eine Thin-Client-Umgebung ist ein Terminalserver als zentrales System für die Anwendungen notwendig. Auf ihm werden alle Anwendungen betrieben, die dem Thin-Client zur Verfügung stehen. Nach der Beschreibung der physischen Trennung X-Client und X-Server und der Optimierung des X-Protokolls durch die NX-Technologie wird nun der Prototyp des Open-Source-basierten Thin-Clients der Stadt Schwäbisch Hall beschrieben.

Ein Thin-Client im Kontext dieses Dokuments besteht aus Hardware, die über einen Prozessor (CPU), einen Arbeitsspeicher (RAM), eine Netzwerkkarte (LAN), eine Grafikkarte (Display) und einen nicht flüchtigen Speicher (*storage*), wie in Abbildung 5 dargestellt, verfügt. Der nicht-flüchtige Speicher kann über verschiedene Anschlüsse, wie USB, IDE oder SD, zur Verfügung gestellt werden.

Von der Softwareseite aus betrachtet, befindet sich auf dem Thin-Client ein Betriebssystem und ein X-Server. Der Thin-Client entspricht einem Arbeitsplatzrechner, wie er in Abschnitt 2 beschrieben wurde. Alle weiteren Anwendungen bezieht der Thin-Client über das Netzwerk von einem zentralen System. Anwendungen werden zentral auf einem oder mehreren Servern ausgeführt und nur die Bildschirmausgabe und die Tastatur- bzw. Mauseingabe werden zwischen Server und Client über das Netzwerk übertragen.

Folgende Vorteile ergeben sich beim Einsatz von Thin-Clients:

- Kleine Desktop-Computer
- Booten über Netzwerk möglich
- Keine Festplatten nötig (sog. „Diskless-Thin-Clients“)

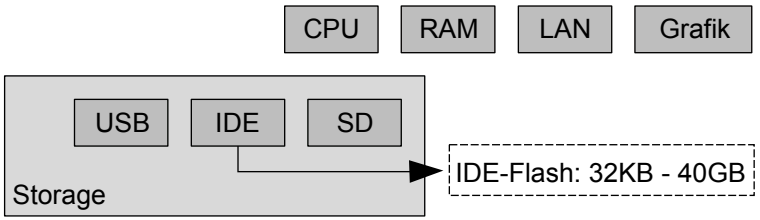


Abbildung 5: Schematische Darstellung der Architektur eines Thin-Clients

- Niedriger Stromverbrauch
- Geringe Wärmeentwicklung
- Geringere Kosten als ein traditioneller Desktop-PC
- Geringe bis keine Geräuschentwicklung (abhängig davon, ob bewegliche Teile wie z. B. Lüfter eingebaut sind)
- Zentrale Administration
- Verwendung von Standardhardware
- Entfernter Zugriff auf Desktop
- Recycling alter PCs zu Thin-Clients möglich

Thin-Clients ohne Storage-Komponente (z. B. Festplatte oder Flashspeicher) werden *Diskless-Thin-Clients* genannt. Sie verfügen über die Komponenten CPU, RAM, LAN und Display und booten ihr Betriebssystem über das Netzwerk vom Terminalserver. Dies geschieht z. B. über *TFTP*. Der Vorteil besteht darin, dass für eine Reihe von Thin-Clients nur ein Betriebssystem-Image auf dem Terminalserver bereitgehalten und gepflegt werden muss. Allerdings ist ein Netzwerk mit entsprechender Bandbreite unbedingt notwendig, da bei jedem Hochfahren eines Thin-Clients das Betriebssystem-Image inklusive X-Server über das Netzwerk übertragen und in den RAM des Thin-Clients geladen werden muss.

Es gibt eine Vielzahl von Hardwareherstellern für Thin-Clients, die jedoch keinen installierten NX-Client im Angebot haben und somit das NX-Protokoll nicht unterstützen. Die Linux-basierten Thin-Clients verfügen über proprietäre Erweiterungen und jeder Hersteller verwendet eigene Lösungen für deren Verwaltung. Es ist kein Management für eine heterogene Thin-Client-Umgebung bekannt. Gemischte Thin–Client-Umgebungen lassen sich somit nicht mit einer Software zentral administrieren. Des Weiteren sind die Clients stark herstellerspezifisch und verwenden keine einheitlichen Software-Images. Die Linux-Systeme verfügen meist noch über einen veralteten Kernel der 2.4er-Reihe.

	LTSP	*PXES*	*Thin Station*
Version	4.1.1	1.0	2.0
Kernelversion	2.6er	2.4er	2.4er
Boot-Methoden	*Etherboot, PXE, RPL*	*PXE*	*Etherboot, PXE*
Notwendige Netzwerkdienste	*DHCP, TFTP*	*DHCP, TFTP*	*DHCP, TFTP*

Tabelle 1: Überblick Thin-Client-Projekte

Neben Hardwareherstellern für Thin-Clients gibt es verschiedene Projekte, die sich dem Thema *server-based computing* auf Softwareseite widmen, von denen die folgenden drei kurz betrachtet wurden:

- *Linux Terminal Server Project (LTSP)*[15]
- *PXES*
- *Thin Station*

Alle drei Projekte bieten Thin-Client-Funktionalität basierend auf *Linux*. Die Seite *Thin-Client.info* beschäftigt sich mit einigen Aspekten des *server-based computing*, mit Thin-Clients und deren Herstellern.[16] Tabelle 1 gibt einen Überblick über die benutzten Kernelversionen und die Boot-Methoden der untersuchten Thin-Client–Projekte. Sie legen den Fokus speziell auf Diskless-Thin-Clients, die ausschließlich über das Netzwerk gebootet werden. Nach Abschluss des Bootvorgangs verbindet sich der X-Server des Thin-Clients mit dem Terminalserver, der den Desktop zur Verfügung stellt. Das LiSoG-Szenario verlangt die Verbindung mit einem NX-Server, der neben dem X-Protokoll auch die Protokolle *RDP* und *VNC* unterstützt. Aber nur das PXES-Projekt unterstützt das NX-Protokoll.

Der Prototyp des Open-Source-basierten Thin-Clients verfügt über eine Storage–Komponente, auf der Betriebssystem und X-Server installiert sind. Somit entfällt das Laden eines Betriebssystem-Images über das Netzwerk beim Booten dieses Thin-Clients. Die Anwendungen befinden sich zentral auf einem Terminalserver, mit dem sich der Thin-Client verbindet.

In Schwäbisch Hall kommen als Storage-Komponenten derzeit IDE-Flash-Speicher zum Einsatz. Sie sind in Größen zwischen 32 KB und 40 GB erhältlich. Der Prototyp verfügt über folgende Hardwareausstattung:

- CPU: *VIA C3* bzw. *Transmeta*, Taktraten zwischen 400 und 800 MHz
- RAM: 128–384 MB
- HD: 1–5 GB
- Grafik: 1024x768, 75 Hz, 32 Bit

15 http://www.ltsp.org
16 http://www.thin-client.info

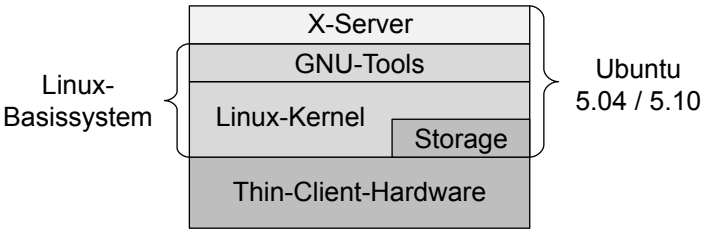


Abbildung 6: Softwarebasis Thin-Client

2.4 Thin-Client-Software

Die Basissoftware des Thin-Clients besteht komplett aus freier Software und Open–Source-Komponenten. Sie umfasst einen Linux-Kernel mit entsprechenden Modulen für die Storage-Komponenten (USB-, IDE- und SD-Unterstützung) und einen X-Server. Der Prototyp verwendet *Ubuntu 5.04* bzw. *5.10* in der vom Installer angebotenen Serverinstallation. Diese Paketierung installiert die wenigsten Pakete. Sie wurde weiter reduziert, so dass das System auf dem Thin-Client nur noch aus wirklich notwendigen Komponenten für den Betrieb der Hardware und den Einsatz des X-Servers installiert ist. Die Installation benötigt ca. 300 MB Speicherplatz auf der Storage-Komponente und kann weiter reduziert werden. Wenn zusätzlich die deutschen Sprachpakete, der Fenstermanager *IceWM*, ein Browser und ein NX-Client installiert werden, benötigt die Installation ca. 450 MB.

Abbildung 6 zeigt den Software-Stack für den Thin-Client. Für die Ausführung von grafischen Programmen ist lediglich ein X-Server notwendig. Ein gestarteter *xterm* im reinen X-Server hat keinen Fensterrahmen oder Symbole zum Maximieren oder Minimieren. Diese Funktionalitäten zum Verändern der Fenstereigenschaften sind im Fenstermanager angesiedelt. Ein traditioneller Linux-basierter Thin-Client verfügt prinzipiell über denselben Software-Stack. Auch die Open-Source-Projekte zum Thema *server-based computing* implementieren den Ansatz aus Abbildung 6. Eine XDMCP-Anfrage[17] an den Fenstermanager des Terminalservers erzeugt einen grafischen Anmeldebildschirm auf dem Bildschirm des Thin-Clients. Der *Display Manager* und der dazugehörige Fenstermanager laufen auf dem Terminalserver.

Mit der Softwarebasis aus Abbildung 6 ergeben sich verschiedene Möglichkeiten, den Thin-Client um Komponenten zu erweitern, wie in Abbildung 7 dargestellt. Der betrachtete Minimalansatz in Abbildung 7 ist mit 1. markiert. Er sieht zusätzlich zur beschriebenen Open-Source-Softwarebasis einen NX-Client vor. Der NX-Client ermöglicht die Verbindung zu einem NX-Server, der mit seiner Proxytätigkeit auf verschiedene Terminalserver mit den Protokollen *X*, *RDP* und *VNC* zugreift. Über den

17 Das *X Display Manager Control Protocol* ist ein Netzwerkprotokoll, das der Kommunikation zwischen einem X-Terminal und einem X-Server dient.

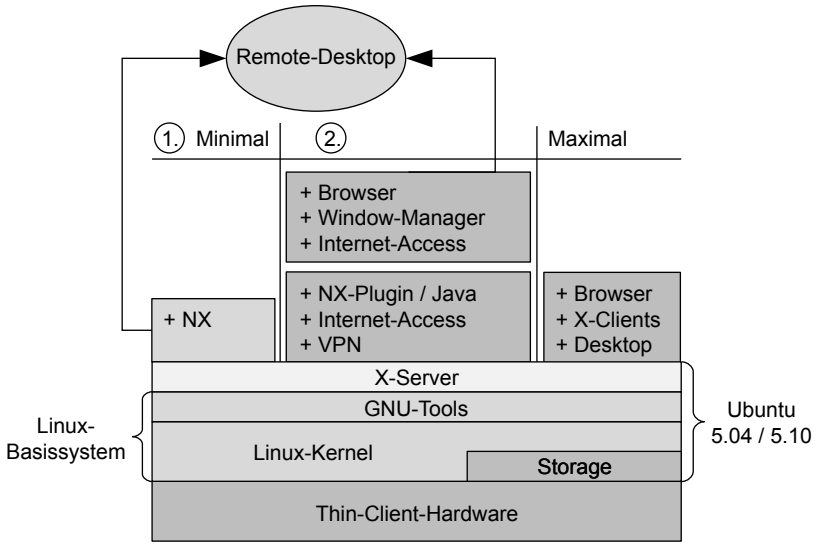


Abbildung 7: Thin-Client-Ansätze

NX-Server können dem Benutzer am Thin-Client verschiedenste Desktop-Systeme zur Verfügung gestellt werden.

Möglichkeit 2 in Abbildung 7 sieht weitere Softwarekomponenten für den Thin–Client vor. Mit Internetzugang, einem Fenstermanager und einem Webbrowser wird aus dem Thin-Client ein einfacher Internetclient, der ohne Zugriff auf einen Terminalserver auskommt und somit auch *offline* arbeiten kann. Mit zusätzlichem NX-Plugin für den Browser und *VPN* ist auch ein Zugriff auf den NX-Server möglich, über den, wie in Möglichkeit 1, ein Remote-Desktop bereitsteht.

Der Maximalansatz hat zusätzlich zu der Softwarebasis ein komplettes Desktopsystem mit Browser und X-Clients. Die Desktop-Anwendungen werden nicht mehr auf dem Terminalserver gestartet und ausgeführt, sondern auf dem Thin-Client selbst. Die Ausstattung moderner Thin-Client-Hardware erlaubt den lokalen Betrieb eines Desktop-Systems, setzt allerdings genügend Speicherkapazität auf der Storage-Komponente voraus.

Die in Abbildung 7 aufgezeigte Möglichkeit 1 wurde auf einem 1-GB-IDE-Flash realisiert. Auf 2 GB kann ein kompletter Desktop installiert werden, zum Beispiel *Ubuntu* oder *Xandros*. Als Storage-Komponente wurden auch 2- bzw. 4-GB-USB–Flashs ausprobiert, auf denen ebenfalls *Ubuntu* installiert wurde. Stehen 5 GB zur Verfügung, so kann sogar ein *SuSE 9.3* auf der Hardware des Thin-Clients installiert werden. Der Prototyp wurde mit *Ubuntu 5.04/5.10* in der Paketauswahl „Server" realisiert, der um die Pakete für einen X-Server erweitert und gemäß der Möglich-

keit 1 aus Abbildung 7 angepasst wurde. Somit wurde ein komplett Linux-basierter Open-Source-Thin-Client bestehend aus Standard-Software-Komponenten beschrieben. Durch die Verwendung von *Ubuntu* steht ein System zur Verfügung, das ausschließlich Open-Source- und freie Software verwendet, einen aktuellen Kernel der Version 2.6.x zur Verfügung stellt, leicht paketierbar und für die Massenproduktion geeignet ist.

3 Anzugehende Problemlösungen

Mit dem Prototyp ist die Grundlage für darauf aufbauende Arbeiten geschaffen. Der nächste Abschnitt zeigt Defizite auf, die aus Sicht der *LiSoG* zu bearbeiten sind.

3.1 Benchmarking

Der Begriff *Benchmarking* (Maßstäbe setzen) bezeichnet ein formalisiertes Konzept, um Verbesserungsmöglichkeiten durch den Vergleich von Leistungsmerkmalen mehrerer vergleichbarer Objekte, Prozesse oder Programme zu finden.

Das Szenario beschreibt den Aufbau eines Open-Source-basierten Thin-Clients und seine architektonische Integration in eine entsprechende Infrastruktur. Ein besonderes Augenmerk liegt auf dem NX-Protokoll und der damit verbundenen Technologie mit ihrem Vorteil der Bandbreiten-schonenden Übertragung, so dass Verbindungen zum NX-Server über GPRS, Modem und ISDN möglich sind. Getestet wurde das Verhalten mit einem Prototyp.

Novell konnte in Berlin die Leistung von Linux-Terminalservern (*SLES9 x86_64*, *KDE*, *OpenOffice.org*, Mail, Browser) im produktiven Betrieb mit mehreren hundert Nutzern beobachten. Basierend darauf können folgende Regeln für die Größenordnung von Linux-Terminalservern genutzt werden (pro 50 *concurrent user*):

- 1 CPU 2.4–3 GHz (*AMD Opteron* 2.4 GHz bzw. *Intel* P4 EM64T 3 GHz)
- 4 GB RAM
- 2–4 GB *swap space*
- 1–2 GB /tmp (abhängig von der Anzahl der Benutzer)

3.2 Hard- und Software

Die Hard- und Software der Thin-Clients muss zusammengestellt werden. Die Software lässt sich bereits leicht über eine spezielle Linux-Distribution paketieren, bei den Hardware-Komponenten mangelt es allerdings noch an passenden Produkten.

Hersteller für Standard-Thin-Client gesucht

Der Prototyp besteht aus Standard-Open-Source-Software-Komponenten. Die Paketierung wird bereits von der verwendeten Linux-Distribution übernommen. Es fehlen

Hersteller, die die beschriebenen Standard-Komponenten als ein Software-Image auf entsprechende Flash-Speicher spielen und anbieten bzw. Hersteller für Thin-Clients, die die Hardware bereits mit der Software-Ausstattung liefern.

Ein Thin-Client-Hersteller kann diesen Thin-Client zusätzlich in sein Produktportfolio aufnehmen und so Kunden bedienen, die explizit vollständig Open Source für den Thin-Client fordern. Solche Kunden können unter anderem Kommunen wie Schwäbisch Hall und öffentliche Einrichtungen sein. Die Paketierung der Standard–Open-Source-Software wird bereits durch die Distribution geleistet. Das benötigte Software-Image kann vom Hersteller selbst erstellt und auf die Hardware aufgespielt werden. Der Mehraufwand in der Produktlinie hält sich hierfür in Grenzen. Das Geschäftsmodell könnte so aussehen, dass der Hersteller Support und Wartung der Geräte anbietet. Der Support der Software wird durch die verwendete Distribution abgedeckt, so dass sich der Support des Herstellers auf die Hardware beschränken lässt.

Systems-Management für Thin-Clients gesucht

Thin-Client-Hersteller bieten für ihre Produkte eine Reihe von eigenen Erweiterungen zur zentralen Verwaltung der Thin-Clients an. Diese Erweiterungen sind meist proprietär und erlauben nur das Management der Hersteller-eigenen Produkte und schließen heterogene Umgebungen aus. Beim Kauf von Thin-Clients möchte sich ein Anwender selten auf einen Anbieter festlegen und seine Infrastruktur mit Thin-Clients von nur einem Hersteller aufbauen lassen. Unter Umständen verfügt der Anwender bereits über Thin-Clients bzw. funktioniert alte PCs zu Thin-Clients um. Damit entsteht eine heterogene Thin-Client-Umgebung, die zentral verwaltet werden soll. Für dieses Szenario reicht die Management-Software eines Thin-Client-Herstellers nicht mehr aus, da zu viele unterschiedliche Architekturen eine Rolle spielen.

Derzeit ist dem Autor keine Open-Source-Software für die zentrale Verwaltung von Thin-Clients, deren Benutzern, Softwareverteilung und Patch-Management bekannt. Es ist allerdings möglich, aus existierenden Bausteinen wie *LDAP*, Open-Source-Datenbanken, Skripten und anderen Werkzeugen ein solches System zu bauen. Benötigt wird eine Lösung, die o. g. Punkte abdeckt. Zu klären sind Details und Anforderungen für ein Systems-Management für Thin-Clients. Die *LiSoG* bearbeitet seit März 2006 das Szenario „Desktop-Management“, aus dem sich einige Punkte des Anforderungskatalogs verwenden lassen, so zum Beispiel zentrales Benutzer-, Gruppen- und Thin-Client- bzw. Computer-Management (*single point of administration*).

Novell bietet beispielsweise mit seinem Produkt *NLPOS* ein Management-Framework für Thin-Clients, das großteils aus Open-Source-Komponenten aufgebaut ist. Auch *Univention* bieten mit seinem *Univention Corporate Server*[18] eine Linux-Komplettlösung, mit der Standort- und Plattform-übergreifend nicht nur Linux-basierte

18 http://www.univention.de/ucs.html

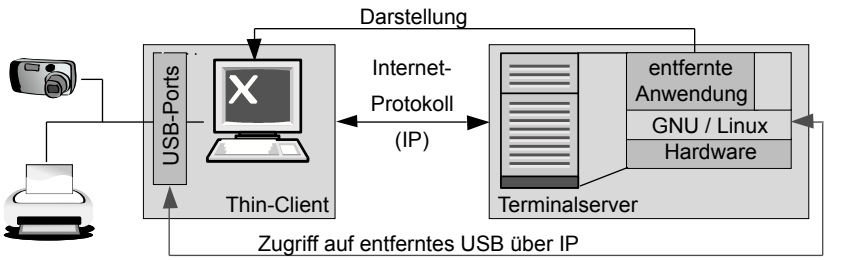


Abbildung 8: USB über Internet-Protokoll (USB over IP)

Systeme, sondern auch heterogene Infrastrukturen verwaltet werden können. Zentrale Komponenten des Managementsystems sind ein LDAP-Verzeichnisdienst und Kerberos-Dienste, so dass sich vorhandene Dienste und Anwendungen sehr einfach integrieren lassen.

3.3 Technische Implementierungen

Die Implementierung neuer Technologien vor allem in Bezug auf die neuen notwendigen USB-Techniken stellen eine große Herausforderung dar.

USB over IP

Das Thema *USB over IP* adressiert die Problematik, lokal an einen Thin-Client angeschlossene USB-Geräte vom Terminalserver aus, wo die Anwendung zur Handhabung von USB läuft, anzusteuern. Beispielsweise soll es möglich sein, einen Drucker oder einen Scanner lokal an den USB-Port des Thin-Clients anzuschließen und über entsprechende Anwendungen des vom Terminalserver bereitgestellten Desktops zu steuern.

Problem hierbei ist die Trennung des USB-Systems von der verarbeitenden Anwendung. Abbildung 8 zeigt die Problemstellung. Soll beispielsweise eine Anwendung auf die Daten einer Digitalkamera, die über USB angeschlossen ist, zugreifen, so erfolgt der Zugriff über das Betriebssystem direkt auf die Hardware. In Abbildung 8 ist die Kamera jedoch an die Hardware des Thin-Clients angeschlossen und die dazugehörige Anwendung läuft entfernt auf einem Terminalserver. Diese Anwendung kann nicht über das Betriebssystem direkt auf die Hardware zugreifen, weil die Digitalkamera nicht an den Terminalserver angeschlossen ist. Die angeschlossene Digitalkamera muss der entfernten Anwendung als lokal angeschlossenes Gerät bekannt gemacht werden, damit der Anwender am Thin-Client damit arbeiten kann.

Das USB/IP-Projekt[19] befasst sich mit dieser Problemstellung. Das Projekt verfügt über einen Entwickler aus Japan und bezeichnet sich selbst als *pre-alpha*.[20] Eine Unterstützung, insbesondere in personeller Hinsicht, könnte das Projekt weiter voranbringen und beschleunigen. Die Funktionalität konnte bisher allerdings noch nicht auf ihre Benutzbarkeit getestet werden.

USB/SIM-Authentifizierung

Mit der USB/SIM-Authentifizierung soll eine Smartcard-ähnliche Authentifizierung vorgenommen werden. Ein USB-Stick besteht hierbei aus einem Flash-Speicher und einem SIM-ähnlichen Chip. Analog zum Konzept der SIM-Karte für Mobiltelefone sind auf dem Chip im USB-Token eine digitale Signatur und ein Zertifikat gespeichert, die den Benutzer des USB-Tokens eindeutig identifizieren. Der USB-Token soll die Funktionalität eines *Security Token* erfüllen. Die Daten auf dem USB-Token sind notwendig zur Authentifizierung des Benutzers gegenüber dem System, im vorliegenden Fall gegenüber einem Thin-Client. Mit dem Zertifikat auf dem Chip sind zusätzlich die Daten auf dem Flash-Speicher verschlüsselt, und der Zugriff darauf wird durch den Chip geregelt. Über eine PIN, ein Passwort oder über biometrische Informationen wird der Zugriff auf die Daten im Flash-Speicher freigegeben.

Vereinfacht gesagt, handelt es sich hierbei um eine Kombination aus *Smartcard* und USB-Stick. Die *Smartcard* dient der Authentifizierung und der USB-Stick der Speicherung von Daten. Die Kombination erlaubt die Verschlüsselung der Daten auf dem USB-Stick, die mit zusätzlichen Informationen, wie PIN, Passwort oder biometrische Informationen, entschlüsselt werden.

Die Idee besteht darin, die Authentifizierung am Thin-Client über ein solches USB-Token zu realisieren. Es gibt Bausteine, mit deren Hilfe eine Lösung zusammengestellt werden kann. Es fehlt jedoch an einem integrierten Produkt. Die Anregung besteht darin, eine solche integrierte Lösung für die Authentifizierung am Thin-Client in einer Linux-basierten Open-Source-Thin-Client-Umgebung umzusetzen, die wiederum Open Source ist. Die Firma *charismathics*[21] vertreibt USB-Sticks mit einer kombinierten Smartcard-Funktionalität.

4 Fazit

Der Thin-Client-Prototyp der Stadt Schwäbisch Hall zeigt, dass die Bestandteile für einen Linux-basierten Open-Source-Thin-Client vorhanden sind. *Linux* und das *X Window System* bieten hervorragende technische Voraussetzungen für eine Thin–Client-Umgebung, die insbesondere bei Standardarbeitsplätzen wie beispielsweise Office mit E-Mail und Internet Sinn macht, da hier keine spezielle Software benötigt

19 http://usbip.naist.jp
20 http://sourceforge.net/projects/usbip
21 http://www.charismathics.com/products/1for_business_customers/3plug--in--icrypt.php

wird und die Hardwareanforderungen nicht besonders hoch sind. Der Arbeitsplatz kann zentral verwaltet werden und verringert somit den Administrationsaufwand.

Wirtschaftlich ermöglicht eine Thin-Client-Umgebung die weitere Nutzung ausgedienter PCs, indem sie zu Thin-Clients umfunktioniert werden. Somit können PCs über fünf Jahre oder länger anstatt der üblichen drei Jahre genutzt werden. Darüber hinaus ist die Thin-Client-Hardware selbst preisgünstiger und wartungsärmer als PC-Hardware.

Durch die NX-Technologie erfährt der Anwender keine Einschränkungen in der Performance der Anwendungen gegenüber einem PC. Der Schritt zum ortsungebundenen Arbeiten ist mit *NX* nicht mehr weit. Wird anstatt eines ausgedienten PCs normale Thin-Client-Hardware eingesetzt, reduziert sich zusätzlich der Geräuschpegel der Hardware.

Der Prototyp der Stadt Schwäbisch Hall zeigt jedoch Defizite bei Linux-basierten Thin-Clients auf. So fehlt es an einem Hersteller oder IT-Dienstleister, der Thin-Clients mit den Software-Images ausliefert und auch Wartung und Support für die Geräte anbietet. Der Software-Support kommt hierbei von der Distribution, im beschriebenen Fall kostenlos von *Ubuntu*.

Des Weiteren fehlt es an einer zentralen Systems-Management-Software-Lösung für Thin-Clients, die verschiedene Thin-Clients auch über Distributionsgrenzen hinweg verwaltet. Auch technisch bietet ein Open-Source-Thin-Client weitere Herausforderungen, wie beispielsweise *USB over IP* oder USB/SIM-Authentifizierung.

Literatur

Bräuner, H. (2005), Linux im Rathaus – Ein Migrationsprojekt der Stadt Schwäbisch Hall, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 37–50. http://www.opensourcejahrbuch.de/download/jb2005/chapter_01/osjb2005-01-04-braeuner [20. Jan 2007].

Bräuner, H. und Pfeifle, K. (2004), 'NX – Turbolader für Linux- und Windows-Terminalserver', Vortrag auf dem Business- & Behördenkongress des LinuxTag 2004. http://www.pl-berichte.de/berichte/lt2004-nxartikel.html [31. Jan 2007].

OpenFacts (2006), 'NX Components', http://openfacts.berlios.de/index-en.phtml?title=NX_Components [17. Jun 2006].

Einbeziehung von Usability-Experten in Open-Source-Community-Projekte: Erfahrungen aus dem OpenUsability-Projekt

ELLEN REITMAYR

(CC-Lizenz siehe Seite 563)

Bis heute verfügen wenige Open-Source-Projekte über eine ausgeprägte Usability-Community[1]. Selbst große Projekte wie die Linux-Desktops *KDE* oder *GNOME* haben nur einen geringen Anteil an fachlich geschulten Usability-Mitwirkenden. In den drei Jahren aktiver Usability-Arbeit im Rahmen des Projektes *OpenUsability*[2] konnten verschiedene Faktoren identifiziert werden, welche die kontinuierliche Zusammenarbeit zwischen Usability und Open-Source-Entwicklung fördern oder behindern können. Diese lassen sich zwei Bereichen zuorden: Zum einen handelt es sich um Voraussetzungen erfolgreicher Usability-Arbeit, die in Open-Source-Projekten oft nicht gegeben sind, zum anderen um strukturelle Eigenschaften der Projekte wie Entscheidungswege oder Integration der Usability-Contributors in die Community. Im folgenden Artikel werden diese Faktoren näher beschrieben und beispielhaft Lösungsansätze aufgezeigt.

Schlüsselwörter: Gebrauchstauglichkeit · Benutzerfreundlichkeit · Usability · Integration · OpenUsability · Community · Open-Source-Projekte

1 Einleitung – Usability im Entwicklungsprozess

„Mit intuitiver Bedienung", „Jetzt noch einfacher", „Einfach wie nie!" – In der Werbung oder in Pressemitteilungen zum Release einer neuen Software gewinnt neben der Funktionalität auch die Einfachheit der Bedienung immer stärker an Bedeutung.

1 Usability (Gebrauchstauglichkeit) ist eine Forschungsrichtung, die Produkte bezüglich ihrer Benutzbarkeit für bestimmte Nutzergruppen und Aufgaben optimiert.

2 Die Plattform *OpenUsability* wurde 2004 gegründet, um Usability und Open-Source-Entwicklung zusammenzubringen: http://www.openusability.org.

Damit ist die Popularität von Usability, also von Methoden, welche die Benutzerfreundlichkeit einer Software erhöhen, in den letzten Jahren stark angestiegen.

Dieser Trend macht auch vor Open-Source-Projekten nicht halt: Mit dem wachsenden Anteil von Open-Source-Software (OSS) auf dem Desktop gilt in vielen Projekten nicht mehr der Grundsatz „von Geeks[1] für Geeks". Die Frage, was der Desktop-Nutzer versteht und was nicht, bleibt jedoch oft reine Spekulation. Um Entscheidungshilfen zu geben, steigt die Nachfrage nach Usability-Experten[2], die User Interfaces nach Nutzungshindernissen durchforsten. Häufig zeigt sich jedoch ein grundsätzliches Missverständnis, was Usability ausmacht. Benutzungsfreundlichkeit wird oft als *yet another feature* wahrgenommen, das einfach zur Liste der Neuerungen beim nächsten Release hinzugefügt werden kann.

Um Software tatsächlich benutzungsfreundlich zu machen, muss Usability jedoch von Anfang an in den Entwicklungsprozess integriert werden. Dieses Prinzip wird in Form der ISO-Norm 13407 *Human-Centred Design Process for Interactive Systems* manifestiert und auch als *usability engineering* bezeichnet. Usability wird hier nicht als punktueller Eingriff definiert, in dem ein bestehendes System auf Nutzungshindernisse überprüft wird, sondern als ein den gesamten Entwicklungs-Zeitraum umspannender Prozess, der mit der Erfassung des Nutzungskontexts und der Definition von Nutzer-Anforderungen noch vor der technischen Planung beginnt. In einem iterativen Prozess werden auf Basis der Nutzer-Anforderungen Design-Entwürfe entwickelt und anhand von Usability-Methoden überprüft. Das Methoden-Repertoire ist an die jeweilige Phase des Entwicklungsprozesses gebunden.[3] Zur Erfassung der Anforderungen werden beispielsweise Aufgabenanalyse oder Personas[4] eingesetzt, in der Phase der Konzeption werden Design-Entwürfe anhand von Papierprototypen verifiziert oder die Informationsstruktur wird mit Hilfe von *Card Sorting* ermittelt. Nutzungshindernisse werden so schon frühzeitig aufgedeckt und können schnell und kostengünstig korrigiert werden.

Die Durchführung einer solchen nutzerorientierten Entwicklung setzt voraus, dass Usability-Spezialisten entscheidend in die Entwicklung einbezogen werden. Ihre Aufgabe ist nicht nur das Design der grafischen Nutzungsoberfläche, sondern auch das Erstellen von Entscheidungshilfen bei der Priorisierung von Funktionalität: Durch Usability-Methoden werden Funktionalitäten identifiziert, die von den Nutzern drin-

1 *Geek* ist ein Anglizismus, der sich aus dem Wort Geck herleiten könnte. Im Zusammenhang mit Computern ist er auch positiv besetzt und bezeichnet hier den engagierten Spezialisten.

2 *Usability-Experte* ist in Deutschland und den meisten Teilen Europas bisher kein Ausbildungsberuf. Die meisten sind Quereinsteiger, die vorher als Informatiker, Ingenieure, Soziologen, Psychologen oder Designer gearbeitet und sich entsprechend weitergebildet haben (Reitmayr et al. 2004). In vielen Studienfächern gibt es heute auch entsprechende Vertiefungen. Für eine Liste an Studienmöglichkeiten siehe Stößel (2005).

3 Für einen Überblick über die Usabilitymethoden im Entwicklungsprozess siehe beispielsweise http://www.usabilitynet.org/tools/methods.htm.

4 Unter einer Persona versteht man daher ein imaginäres Modell einer Person mit allerdings sehr konkreten (Charakter-)Eigenschaften oder etwa Nutzungsverhalten.

gend benötigt werden, und es wird überprüft, wie diese Funktionalitäten am besten in die bestehende Software integriert werden können.

Das Bewusstsein von Usability als *Prozess* fehlt sowohl in der proprietären als auch in der Open-Source-Entwicklung noch zu großen Teilen. In der proprietären Software-Welt ist Usability oft ein externer Dienstleister, der zwar zu bestimmten Zeitpunkten den Pfad der Entwicklung kreuzt, jedoch wenig konzeptionelle Arbeit leistet. Erfreuliche Ausnahmen bilden hier Groß-Firmen wie *Microsoft*[5], *SAP*[6] oder *IBM*[7], die schon seit Jahren *user-centered design* propagieren und durch eigene Abteilungen aktiv betreiben.

2 Usability in Open-Source-Projekten

Bis heute verfügen wenige Open-Source-Projekte über eine ausgeprägte Usability–Community. Zwar haben Nutzer, Entwickler und andere Interessenvertreter die Möglichkeit, Kommentare und Verbesserungswünsche in Usability-Mailinglisten oder -Foren zu diskutieren, doch ist der Anteil der fachlich geschulten Mitwirkenden verschwindend gering. Selbst große Projekte wie die Linux-Desktops *KDE* und *GNOME* verfügen nur über einen geringen Anteil an ausgebildeten Usability-Studenten oder -Professionals.

Das geringe Engagement von Usability-Experten in Open-Source-Projekten scheint auf den ersten Blick verwunderlich: Wie Mühlig (2005) beschrieb, kann die Usability–Arbeit in OSS-Projekten äußerst erfüllend sein. Da individuelle Entwickler oft über hohen Einfluss verfügen, können substanzielle Änderungen aus Usability-Empfehlungen zeitnah und ohne Diskussionen umgesetzt werden – ein Erfolgsgefühl, das man derart in kommerziellen Projekten selten erfährt. Auch eignen sich Open-Source–Projekte sehr gut zur Aus- und Weiterbildung, da selten Grenzen durch Zeitdruck gesetzt sind und keine Vertraulichkeits-Vereinbarungen getroffen werden müssen.

Eventuell sind die Schwellen bei der Annäherung zu hoch: Welches Open-Source–Projekt soll man als Usability-Spezialist unterstützen, wie kann man es ansprechen? Wie kommt man als Open-Source-Projekt an kompetente Usability-Leute? Um diesen Erst-Kontakt zu erleichtern, wurde 2004 die Plattform *OpenUsability*[8] gegründet, die interessierte Open-Source-Projekte und Usability-Spezialisten zusammenbringt.

Die Plattform erlebte seit ihrer Gründung – sowohl auf Seite der Entwickler als auch der Usability-Spezialisten – einen starken Zuspruch, der sich unter anderem in der Zahl an Registrierungen oder Pressemeldungen widerspiegelte. Trotzdem wird bis heute nur ein Bruchteil der rund 170 registrierten Projekte (Stand: Oktober 2006) aktiv betreut. Während hierfür zum Teil Schwächen der Plattform verantwortlich

5 *Microsoft User Interface Design and Development*: http://msdn.microsoft.com/ui/.
6 *SAP Design Guild*: http://www.sapdesignguild.org.
7 *IBM Ease of Use*: http://www-03.ibm.com/easy/.
8 http://www.openusability.org

sind – Usability-Spezialisten werden nicht in ausreichender Weise über interessante Projekte oder offene Jobs informiert – stellte sich zusätzlich heraus, dass unter bestimmten strukturellen Bedingungen Probleme bei der Kooperation zwischen Usability und Open-Source-Entwicklung auftreten, die dazu führen können, dass die Zusammenarbeit frühzeitig im Sande verläuft.

In den drei Jahren aktiver Usability-Arbeit im Rahmen des Projektes *OpenUsability* konnten wir Faktoren identifizieren, welche die fortlaufende Zusammenarbeit zwischen Usability und Entwicklung in Open-Source-Projekten beeinflussen. Diese beschreiben zum einen Voraussetzungen der Usability-Arbeit in Open-Source-Projekten, zum anderen strukturelle Eigenschaften in der Community.

2.1 Voraussetzungen der Usability-Arbeit in Open-Source-Projekten

Ein wesentliches Mantra der Usability ist der Leitsatz „Know your User". Denn ohne Kenntnis, wer eine Software später verwenden wird, welche Anforderungen er hat und welche Aufgaben er mit der Software bestreiten wird, ist eine nutzerorientierte Entwicklung, wie in der Einleitung beschrieben, nicht möglich.

Beschreibung der Zielgruppe

In Open-Source-Projekten ist die Frage der Zielgruppe oft nicht eindeutig geklärt. Fragt man Entwickler, für wen sie Software entwickeln, erhält man selten klare Aussagen. So wird häufig der wenig spezifische Durchschnittsnutzer oder *Jedermann* genannt (z. B. Trillitzsch 2006). Der Gedanke des *freien Quellcodes* wird hier auf die Nutzerschaft übertragen: Was für alle frei zugänglich sein soll, soll auch für alle gleichermaßen zur Nutzung bereitstehen. Außer Acht wird dabei gelassen, dass die vielzitierte „Tante Tillie" (z. B. Raymond 2004) und der „Geek", also der professionelle Softwareentwickler, sehr unterschiedliche Ansprüche an eine Software haben. Für den Usability-Betreuer ist es schwierig, die Software auf den Endnutzer abzustimmen, da je nach Fragestellung die eigenen Bedürfnisse, die „des Powerusers" oder „meiner Oma" angeführt werden.[9]

Auch im GIMP-Projekt[10], einer Software zur Manipulation von Bildern, zeichnete sich dieser Trend ab: Als lange Zeit einzige ernstzunehmende freie Bildbearbeitungs–Software versuchte der *GIMP*, die gesamte Bandbreite an potenziellen Nutzern zu

9 Ein anschauliches Beispiel bietet eine Diskussion über die Usability von Suchdialogen in der Mailingliste *kde-core-devel* unter http://lists.kde.org/?t=114767296400002&r=1&w=2. In insgesamt 41 E-Mails diskutierten die Entwickler über mögliche Verbesserungen des Suchdialogs. Eine Streitfrage bildete die Suche nach regulären Ausdrücken, die von einigen Entwicklern selbst benötigt und damit stark verteidigt wurde. Andere argumentierten mit dem *average user* oder der *grandma*, und eine Stimme behauptete sogar, dass *KDE* gar nicht für Entwickler gemacht werden solle. Die Diskussionen drehten sich im Kreis, da die Beteiligten meist auf ihren Standpunkt beharrten. Welche Implementierung schließlich umgesetzt wurde, bleibt im Thread offen.

10 *GIMP* steht für das *GNU Image Manipulation Program*.

bedienen. Diese reichte von Gelegenheitsnutzern, die beispielsweise die Bildgröße oder den Rote-Augen-Effekt eines Fotos manipulieren wollten, bis hin zum professionellen, künstlerischen Einsatz.

Die fehlende Ausrichtung auf eine definierte Zielgruppe resultierte in einem unkontrollierten Zuwachs an Funktionalität, vor allem im Bereich der Filter und Erweiterungen. Selbst für geübte Nutzer wurde es immer schwieriger, einen Überblick über die Funktionalität zu erhalten. Mehrere Usability-Berater, die im Rahmen des Projektes *OpenUsability* über mögliche Lösungsansätze bezüglich der Usability des *GIMP* diskutierten, drehten sich immer wieder im Kreis, wenn es zu der Frage kam, welche Funktionalitäten eine Überarbeitung benötigten.

In einer Sitzung zur Definition der Zielgruppe des *GIMP* bei der *Libre Graphics Conference* in Lyon 2006 wurde die Zielgruppe schließlich klar auf High-End-Nutzer festgelegt (Joost und Quinet 2006). Diese richtungsweisenden Vorgaben erlauben nun die Priorisierung der Schritte bei der anstehenden Überarbeitung der Informationsarchitektur und des Interaktionsdesigns des *GIMP*.

Repräsentative Nutzung des user-feedback

Aber Usability-Arbeit bedeutet nicht zwangsläufig wie beim *GIMP* eine Neudefinition der bestehenden Nutzergruppe. Oft fehlt in Open-Source-Projekten lediglich eine klare und übersichtliche Erfassung von Nutzer-Bedürfnissen.

Bug-Tracking-Systeme oder User-Foren erlauben zwar prinzipiell jedem Nutzer, seine Bedürfnisse zu kommunizieren, doch werden sie oft nur von einem bestimmten Teil der Gesamt-Nutzerschaft verwendet. Auch fühlen sich Entwickler oft überfordert von der Menge an Einträgen, so dass sie zu vorschnellen Urteilen beim Akzeptieren oder Ablehnen von Anfragen neigen: In einer Analyse der Bugtracking-Systeme verschiedener GNOME-Applikationen verglich Yeats (2006) die Reaktionen von Entwicklern auf Feedback von sogenannten Code-Savvy-Nutzern – also Bug-Reports, die Wissen über die Quellcode-Struktur erkennen ließen – mit Reaktionen auf Bug-Reports ohne Bezug zum Quellcode. Die Kommentare der *normalen Nutzer* wurden meist zugunsten der Code-Savvy-Kommentare vernachlässigt.

Auch in Nutzer-Foren werden *feature requests* von *normalen Nutzern* häufig vorschnell anhand technischer Maßstäbe bewertet (Yeats 2006). Als Resultat greifen *user innovation cycles* (von Hippel 2005) nicht mehr. Oft werden lange nicht alle innovativen Ideen, die aus kontinuierlichen Rückmeldungen von Nutzern der eigenen Software gewonnen werden könnten, auch tatsächlich aufgegriffen. Der Erfolg des Apache-Servers gegenüber proprietären Alternativen ist laut von Hippel beispielsweise maßgeblich auf derartige Feedback-Zyklen zurückzuführen.

Sinnvoller wäre folglich eine Bewertung anhand der Gewichtung der Einträge: Ausschlaggebend sollte sein, ob ein *feature request*, also der Wunsch nach zusätzlicher Funktionalität, von einer Einzelperson gestützt wird oder ob eine Vielzahl an Nutzern dahintersteht.

Um Bewertungen, Meinungen und *feature requests* zusammenzuführen, eignen sich repräsentative Umfragen innerhalb der Nutzer-Basis. Bei zahlreichen Projekten, die auf *OpenUsability* registriert sind, konnten Umfragen helfen, Features zu priorisieren und so die Entwicklung in die Richtung der Nutzer-Bedürfnisse zu lenken, statt sie durch technische Machbarkeit oder Herausforderung leiten zu lassen.

2.2 Strukturelle Eigenschaften der Community

Strukturelle Eigenschaften der Community kommen vor allem in großen Projekten zum Tragen, in denen die Zusammenarbeit nicht mehr ohne ein gewisses Maß an Management und Organisation funktioniert. Hier handelt es sich nicht nur um OSS–spezifische Merkmale. Auch proprietäre Software-Entwicklung kämpft mit ähnlichen Problemen, wenn Kommunikationspfaden und Wissensmanagement beim Wachstum zu wenig Beachtung geschenkt wurde.

Trotzdem treten gewisse strukturelle Merkmale, die Auswirkungen auf die Usability-Arbeit haben, in OSS-Projekten häufiger auf als in proprietären Projekten. Eines dieser Merkmale ist die hierarchische Struktur, die häufig mit dem Erfolg einer Open–Source-Software korreliert (Healy und Schussman 2003).

Auch in der Usability-Arbeit stellte sich dieser Faktor als relevant heraus: Fehlende hierarchische Entscheidungswege stellen in großen Projekten Probleme bei der Zusammenarbeit dar. Möchte ein Usability-Spezialist hier Vorschläge umsetzen, muss er jeden einzelnen Entwickler von den Vorteilen überzeugen, statt eines einzelnen *project leaders*[11].

Dass Entwickler in Open-Source-Projekten normalerweise *das letzte Wort* haben, bezeichnete Patricia Jung (2006) als das „Primat der Programmierer“: Diejenigen, die den Code schreiben, tragen die Entscheidung über Features und Gestaltung des User Interfaces. Dieses „Primat“ ist unproblematisch, so lange die Entwickler gegenüber Usability aufgeschlossen sind und Vorschläge ernsthaft diskutieren und umsetzen. Sind einzelne Entwickler jedoch skeptisch oder gar abgeneigt, so kann eine ganzheitliche Umsetzung von Änderungen am User Interface nicht mehr garantiert werden.

Im Projekt *KDE*, einer Desktop-Umgebung für *Linux*, zeigen sich die Auswirkungen hierarchischer Entscheidungswege und des „Primats der Programmierer“ besonders deutlich. Durch seine Größe und Struktur stellt die Summe der KDE–Teilprojekte ein gutes Vergleichsstudienobjekt dar, da es neben dem *Core* aus einer Vielzahl an kleineren, unabhängigen Software-Projekten besteht. Eine konsistente hierarchische Struktur ist zwischen den einzelnen KDE-Teilprojekten nicht zu beobachten. Während manche Projekte ein strenges Reglement dahingehend aufweisen, welcher Entwickler welchen Code beitragen darf, sind andere offen. Dies hat auch Auswirkungen auf die Möglichkeiten von Usability-Spezialisten und ihre Bereitschaft, sich aktiv an Open-Source-Projekten zu beteiligen.

11 Ein *leader* ist im Sinne von Healy und Schussman (2003) ein Kern-Entwickler, dessen Ansichten respektiert werden und dessen Vorgaben der Rest des Projektes folgt.

In den folgenden Abschnitten werden die Begebenheiten bezüglich der Beobachtungen von Jung (2006) und Healy und Schussman (2003) in einzelnen KDE-Teilprojekten näher beschrieben und der Zusammenhang zur Integration von Usability–Methoden veranschaulicht.

Die Rolle der Usability-Befürworter im Projekt

Zunächst betreuten verschiedene OpenUsability-Berater einzelne KDE-Projekte wie *Kontact*, *Kivio*, *DigiKam* oder *Kuroo*. Jedes dieser Projekte umfasste eine überschaubare Anzahl an Entwicklern, zudem hatten jeweils die einflussreichen Kern-Entwickler – *leader* im Sinne von Healy und Schussman (2003) – um Usability-Beratung gebeten, waren der Thematik gegenüber also aufgeschlossen. Die Zusammenarbeit wurde von beiden Seiten als fruchtbar bewertet.

Anders der Beginn der OpenUsability-Aktivität im Projekt *Amarok*, einem Audio–Player für *KDE*: Hier bat ein einzelner Entwickler um Usability-Beratung, ohne die Kern-Entwickler des Projektes darüber zu informieren. Ein Usability-Spezialist führte bald darauf eine Analyse durch und leitete die Ergebnisse an das Projekt weiter. Die Core-Developer, die ja nicht selbständig um Usability gebeten hatten, fühlten sich durch den Report angegriffen und beachteten ihn nicht weiter.

Ein Grund, der zu dieser enttäuschenden Reaktion führte, war die fehlende Absprache mit den Projektleitern. Wie in der proprietären Software-Entwicklung auch kann Usability nur effektiv umgesetzt werden, wenn die einflussreichen Stellen von der Idee überzeugt sind und Usability aktiv fördern.

Zur Verteidigung der Amarok-Entwickler muss eingeräumt werden, dass der Usability-Spezialist im Vorfeld nicht mit dem Projekt über dessen Ziele und Zielgruppe gesprochen hat: Während der Report generelle Richtlinien berücksichtigte, hatten die Amarok-Entwickler damals eine besondere Vorstellung, wie sich ihre Software verhalten sollte. Eine klare Absprache über diese Ziele vor Beginn der Usability-Aktivität hätte Missverständnisse verhindert. Erfahrungen aus anderen Projekte stützen diese Annahme.

Kontinuierliches Engagement der Usability-Spezialisten

Punktuelle und generelle Reports sind die einfachste Methode, Usability-Feedback an ein Projekt zurückzumelden. Usability-Spezialisten können den Zeitaufwand für das Erstellen des Reports leicht einschätzen, mit etwas Extra-Zeit für die Diskussion ist der Beitrag für sie zügig erledigt. In der Praxis hat sich jedoch gezeigt, dass Entwickler unterschiedliche Prioritäten bei der Implementierung der Vorschläge setzen. Statt sie im Nutzungs-Kontext zu beurteilen, werden *quick fixes* in der Regel zuerst implementiert, selbst wenn im Report andere Prioritäten gesetzt sind. Bei *Kontact*, dem *KDE Personal Information Manager*, führte dies zu Inkonsistenzen in der Menüstruktur

der Sub-Komponenten Mail, Adressbuch und Kalender, da Empfehlungen für einige Teile umgesetzt wurden, für andere nicht.

Insgesamt hat sich gezeigt, dass punktuelle Reports anhand genereller Usability–Heuristiken lediglich in einer Vielzahl an *quick fixes* resultieren. Die häufig notwendigen, umfangreicheren Änderungen am Interaktions-Design oder der Informations–Architektur werden jedoch selten umgesetzt. Auch dies gilt in gleicher Weise für die proprietäre Software-Entwicklung – punktuelle Reports ohne weitere Anstrengungen widersprechen der Idee eines nutzerorientierten Entwicklungsprozesses (vgl. Abschnitt 1).

Dennoch: Punktuelle Reports sind besser als kein Feedback und können als Einstiegspunkt für langfristige Usability-Arbeit dienen. Dass dies häufig nicht der Fall ist, lässt sich wiederum auf das „Primat der Programmierer" (Jung 2006) und die von Yeats (2006) beschriebenen Reaktionen auf nicht-technisches Feedback zurückführen: Beiträge von Usability-Mitwirkenden, die sich innerhalb der Community noch keinen Namen gemacht haben, werden häufig vorschnell mit einem *won't fix* abgetan. Nur hochmotivierte Usability-Spezialisten werden sich dann auf weitere Diskussionen einlassen.

Im KDE-Projekt wurde deshalb versucht, die Usability-Mitwirkenden durch Blogs und Interviews stärker in die Community einzubinden. Auf der einen Seite erhöhte dies die Bindung und damit das kontinuierliche Engagement der Usability-Berater, auf der anderen Seite wurde unter den Entwicklern das Vertrauen in Usability gestärkt. Die Usability-Experten waren keine Außenstehenden mehr, die ungefragt die eigene Software kritisieren, sondern wurden zu gleichwertigen Mitgliedern des Projektes.

Verantwortlichkeiten und Entscheidungswege

Mit steigendem Engagement im Projekt realisierten die Usability-Berater in *KDE* bald, dass sie immer wieder die gleichen Probleme an unterschiedlichen Stellen des Projektes angingen. So wurden ähnliche Interaktions-Elemente benötigt, die jeweils in einem Unterprojekt neu implementiert wurden.[12] Auch erläuterten sie gegenüber einzelnen Programmierern stets die gleichen Usability-Grundsätze, was viel Zeit und Energie kostete.

Es zeigte sich, dass dem KDE-Desktop eine einheitliche Richtung fehlte. Die Usability-Spezialisten sahen sich einer neuen Herausforderung gegenübergestellt: Um Änderungen konsistent in *KDE* umsetzen zu können, wurden allgemeine *guidelines* sowie Änderungen an den Bibliotheken der Desktop-Umgebung notwendig.

Um dies zu erreichen, ist eine enge Zusammenarbeit mit den Entwicklern ausschlaggebend. An den Bibliotheken ist jedoch eine Vielzahl Entwickler beteiligt, kla-

12 Eine optimierte Darstellung der Zoom-Funktion auf dem Toolbar wurde beispielsweise in einem Office-Programm, dem PDF-Viewer und einem weiteren Programm jeweils einzeln implementiert. Sinnvoller wäre eine Manifestierung der Optimierung in den Bibliotheken, doch fand sich hierfür bisher kein Programmierer.

re Verantwortlichkeiten fehlen. Die Implementierung von Usability-Verbesserungen scheiterte somit oft daran, dass die Usability-Berater zunächst selbständig einen Entwickler finden mussten, der Zeit und Fähigkeiten besitzt, die Änderung umzusetzen. Mailinglisten stellten sich als ein wenig geeignetes Medium heraus, um Entwickler zu werben, da Requests oft ungehört im Archiv untergingen.

Um diesen Problemen zu begegnen, wurden Ende 2005 verschiedene Umstrukturierungen in der KDE-Community durchgesetzt.[13] Mit der Schaffung einer technischen Arbeitsgruppe[14] wurde das Problem der fehlenden Verantwortlichkeiten angegangen, so sollten klare Ansprechpartner für bestimmte technische Bereiche geschaffen werden. Für Anfang 2007 sind Job-Listen geplant, auf denen unter anderem Requests des Usability-Teams permanent ausgeschrieben sind. *Human-interface guidelines*[15] als allgemein verbindliche Richtlinien bei der Gestaltung von Anwendungen sind in der Entstehung.

3 Fazit – Integration in die Community

Entscheidet sich eine Community, nicht (nur) für sich *selbst*, sondern für Nutzer mit anderen Bedürfnissen zu entwickeln, kann der Grundsatz *just for fun* nur noch bedingt gelten. Wie in der proprietären Softwareentwicklung ist Usability in Open-Source–Projekten kein Ziel, das man von heute auf morgen umsetzen kann. Die Einführung von *Usability-Engineering* bedeutet eine klare Definition von Zielstellungen, auf welche die Software hin *maßgeschneidert* wird, sowie strukturelle Änderungen, bei denen Usability-Spezialisten zu gleichberechtigten Mitgliedern im Projekt werden.

Insbesondere der letzte Punkt, also die kontinuierliche Integration von Usability in den Open-Source-Entwicklungsprozess, stellt sich immer wieder als problematisch heraus. Wird Usability nicht ausreichend von einflussreichen Entwicklern propagiert und aktiv gefördert, bleiben Verbesserungsvorschläge aus *usability reports* häufig unbeachtet. Werden diese *reports* von Usability-Spezialisten erstellt, die im Projekt bisher unbekannt sind, finden die *reports* leider noch weniger Aufmerksamkeit – als Konsequenz wenden sich diese Usability-Spezialisten dann anderen Aufgaben zu.

Doch selbst, wenn einflussreiche Entwickler Usability befürworten, ist ihre Umsetzung in der Software noch nicht garantiert. Vor allem in Projekten, in denen klare Entscheidungswege und Entwicklungs-Richtlinien fehlen, muss dann bei jedem einzelnen Entwickler ein Bewusstsein für die Anforderungen der Usability vorhanden sein. Eine Verankerung von Usability-Zielstellungen, etwa mittels einer *Vision* oder durch Festschreibung von Richtlinien, sowie *Aufklärungsarbeit* in Blogs, Artikeln

13 Die Änderungen wurden in der jährlichen Vereinssitzung des *KDE e. V.* mehrheitlich beschlossen (siehe http://ev.kde.org/meetings/2005.php).

14 Siehe http://dot.kde.org/1137180087/ und http://dot.kde.org/1139614608/.

15 Siehe http://wiki.openusability.org/guidelines.

oder Workshops zeigte gute Erfolge – und ist somit ein weiterer Schritt hin zu dem Grundsatz der Anwenderorientierung.

Literatur

Healy, K. und Schussman, A. (2003), The Ecology of Open-Source Software Development, Working Paper, Department of Sociology, University of Arizona.

Joost, R. und Quinet, R. (2006), 'The GIMP Developers Conference 2006 – GIMP vision', http://developer.gimp.org/gimpcon/2006/index.html#vision [17. Dez 2006].

Jung, P. (2006), Frauen-freie Zone Open Source?, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 235–250. http://www.opensourcejahrbuch.de/download/jb2006/chapter_05/osjb2006-05-03-jung [17. Dez 2006].

Mühlig, J. (2005), Open Source und Usability, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 87–94. http://www.opensourcejahrbuch.de/download/jb2005/chapter_02/osjb2005-02-02-muehlig [17. Dez 2006].

Raymond, E. S. (2004), 'The Luxury of Ignorance: An Open-Source Horror Story'. http://www.catb.org/~esr/writings/cups-horror.html [17. Dez 2006].

Reitmayr, E., Vogt, P., Beu, A., Mauch, D. und Röse, K. (2004), Branchenreport und Honorarspiegel 2003, Bericht, German Chapter der Usability Professionals' Association e.V., Stuttgart. http://www.gc-upa.de/files/upa_report_fn.pdf [17. Dez 2006].

Stößel, C. (2005), 'World Usability Day 2005: Informationsbroschüre Studienmöglichkeiten', German Chapter der Usability Professionals' Association e.V. http://www.gc-upa.de/files/UsabilityStudiengaenge.pdf [17. Dez 2006].

Trillitzsch, T. (2006), KDE Target Users: The Various Comments about the Intended KDE Audience. Unpublished Analysis of the 'People behind KDE' Interview Series.

Yeats, D. (2006), Open-Source Software Development and User-Centered Design: A Study of Open-Source Practices and Participants, PhD thesis, Texas Tech University, Lubbock, TX. http://etd.lib.ttu.edu/theses/available/etd-07232006-221754/unrestricted/Yeats_Dave_Diss.pdf [17. Dez 2006].

von Hippel, E. (2005), „Anwender-Innovationsnetzwerke": Hersteller entbehrlich, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 449–462. http://www.opensourcejahrbuch.de/download/jb2005/chapter_07/osjb2005-07-04-vonhippel [17. Dez 2006].

Turaya – Die offene Trusted-Computing-Sicherheitsplattform

NORBERT POHLMANN UND MARKUS LINNEMANN

(CC-Lizenz siehe Seite 563)

Mit Turaya stellt die Forschungs- und Entwicklungsgemeinschaft des Projekts *European Multilaterally Secure Computing Base (EMSCB)* die erste vertrauenswürdige, faire und offene Sicherheitsplattform zur Verfügung, die auf Trusted-Computing-Technologie aufbaut. Sie bietet eine derart hohe Sicherheit, dass durch sie Möglichkeiten für innovative Geschäftsmodelle und kreative Anwendungen geschaffen werden können. Das als manipulationssichere Komponente in der Hardware eingesetzte „Trusted Platform Module" (TPM) ermöglicht die zuverlässige Überprüfung der Integrität und Authentizität eines IT-Systems – auch aus der Entfernung. Der Open-Source-Ansatz bietet die Möglichkeit, die viel diskutierte Trusted-Computing-Technologie vertrauenswürdig einzusetzen.

Schlüsselwörter: EMSCB · Turaya · Trusted Platform Module

1 Einleitung

Die Zahl der Angriffe auf Computersysteme durch Viren und Trojanische Pferde nimmt stetig zu. Der Zeitraum für die Ausnutzung von gerade bekannt gewordenen Schwachstellen in Softwaresystemen liegt derzeit bei 6,4 Tagen (Barnitzke 2005) und sinkt weiter. Des Weiteren nimmt die Komplexität der Angriffe immer weiter zu (Windows IT Pro 2004), während gleichzeitig immer weniger Kenntnisse notwendig sind, um solche Angriffe durchzuführen. Auch die starke Verbreitung von Bot-Netzen[1] stellt eine nicht zu unterschätzende Bedrohung für die Betroffenen dar. Angreifer durchbrechen die vorhandenen Sicherheitsmechanismen der Software- und

1 Bot-Netze sind Netzwerke, bestehend aus Rechnersystemen, die ohne Wissen der Besitzer unter die Kontrolle von Außenstehenden gebracht wurden und beispielsweise Spam-Mails versenden oder Denial-of-Service-Angriffe durchführen.

Betriebssysteme, und es stehen zur Zeit keine schärferen Sicherheitsmaßnahmen zur Verfügung, die dieser Problematik entgegenwirken könnten. Auch zusätzliche konventionelle Sicherheitslösungen wie Firewalls oder Intrusion-Detection-Systeme (IDS)[2] bieten diesbezüglich oft keinen ausreichenden Schutz. Die statischen Infrastrukturen einzeln stehender Systeme sind in den vergangenen Jahren von heterogenen und dynamischen Netzen verdrängt worden. Dabei verschwammen die einst klaren Firmen- und Systemgrenzen, die mit herkömmlichen Sicherheitsmechanismen, wie z. B. Firewalls abgesichert werden konnten.

Was wäre wenn:

- Sie ein vertrauliches Dokument verschlüsselt an einen Kollegen sendeten, der es auf dem Flurdrucker ausdruckt und vergisst? Für solche Fälle hätten Sie gerne die Möglichkeit gehabt, das Drucken Ihres Dokuments auf fremden Rechnersystemen zu untersagen!
- Ein Trojanisches Pferd einen *keylogger* auf Ihrem Rechnersystem installierte und all Ihre Passworteingaben mitläse? Dann wollten Sie, dass Ihre sicherheitsrelevanten Daten nicht unter dem Zugriff des Betriebssystems stünden!
- Ein Schädling die privaten Schlüssel Ihrer Zertifikate ausläse und versendete? Dann würden Sie einen sicheren Zertifikatsspeicher herbeisehnen!
- Sich jemand als externer Mitarbeiter mit dem *PDA* an Ihrem Firmenserver anmeldete und gar nicht Ihr Mitarbeiter wäre? Dann wünschten Sie, dass Sie das externe Rechnersystem mit hoher Vertrauenswürdigkeit authentifizieren könnten!
- Sie ein Sicherheitssystem einer Firma einsetzten und herausfänden, dass es eine Hintertür eingebaut hätte? Dann würden Sie beim nächsten Mal auf ein Rechnersystem zurückgreifen, welches eindeutig und offen evaluiert werden könnte!

Die dargestellten Szenarien unterstreichen die Motivation zur Realisierung einer vertrauenswürdigen, fairen und offenen Sicherheitsplattform, die im Rahmen dieses Artikels vorgestellt werden soll.

2 Anforderungen und Ziele

Daten besitzen einen hohen Wert und sollten daher mit angemessenen Sicherheitsmechanismen geschützt werden. Um auf private und vertrauliche Daten zugreifen zu können, ist folglich eine Authentifizierung notwendig. Diese kann auf unterschiedliche Arten durchgeführt werden und reicht vom Passwortschutz bis zur Zwei-Faktor–

2 Intrusion-Detection-Systeme sind Programme, deren Aufgabe darin besteht, Einbrüche in Rechnersysteme zu detektieren.

Authentifizierung[3], z. B. mit einer *SmartCard*. Ein allen gemeinsames Sicherheitsproblem bei diesen herkömmlichen Verfahren ist die Allmächtigkeit des Betriebssystems. Sobald das Betriebssystem kompromittiert wird, hat der Angreifer die volle Kontrolle über jegliche Aktion und damit auch über sämtliche Daten, zu denen auch Zugangsdaten, Zertifikate und Schlüssel gehören. Durch die Strukturen der am häufigsten eingesetzten Betriebssysteme ist eine „echte" Sicherheit nicht möglich. Dies verdeutlicht allein die Vielzahl an Sicherheits-Patches, die fortlaufend herausgegeben werden, um entdeckte Sicherheitslücken zu schließen. Dies lässt den Schluss zu, dass neue Sicherheitskonzepte notwendig sind, um eine höhere Sicherheit und Vertrauenswürdigkeit zu erreichen und trotzdem die Arbeitsfähigkeit herkömmlicher Rechnersysteme zu erhalten. Die erste Anforderung an ein vertrauenswürdiges Rechnersystem[4] ist somit, dass alle bisherigen Anwendungen weiterhin genutzt werden können und zusätzlich alle sicherheitskritischen Daten geschützt werden, indem diese nicht mehr unter der Kontrolle eines herkömmlichen Betriebssystems stehen.

Heutige IT-Anwendungen beschränken sich nicht mehr nur auf reine PC- und Server-Systeme von Unternehmen. Neue Geschäftsmodelle erfordern die Realisierung innovativer Anwendungen zum Beispiel auch für eingebettete Rechnersysteme. Dazu zählen mobile Geräte, wie *PDA*s und *Smartphones*, aber auch Rechnersysteme im Automobilbereich wie Leistungssteuerungs- und Infotainment-Systeme. Die zu entwickelnde Sicherheitstechnologie muss den Anspruch erfüllen, auf einem breiten Spektrum von Rechnersystemen einsetzbar und von diesen unabhängig zu sein.

Verteilte Anwendungen fordern die Vernetzung voneinander entfernter Rechnersysteme. Ein externer Mitarbeiter einer Firma greift etwa mit einem mobilen Gerät auf den gesicherten Firmenserver zu. Ist das System des mobilen Gerätes jedoch kompromittiert, so ist der Server nun ebenfalls in Gefahr. Wenn Zugangsdaten durch kriminelle Handlungen oder auch durch die Unachtsamkeit des externen Mitarbeiters in die falschen Hände geraten sind, dann ist auch der Server nicht mehr als sicher einzustufen. Einem Angreifer ist im oben beschriebenen Fall die Möglichkeit gegeben, von einem fremden Rechnersystem auf den Server zuzugreifen. Um in solchen Szenarien wirklich eine vertrauenswürdige Verbindung etablieren zu können, muss sich demnach nicht nur der Anwender, sondern auch das Rechnersystem mit all seinen Hard- und Softwarekomponenten und seiner Konfiguration authentifizieren und das System muss zusätzlich garantieren können, dass es nicht kompromittiert worden ist.

Wünschenswert erscheint also eine Sicherheitsplattform, die die verwendete Soft- und Hardwarekonfiguration verlässlich für den Benutzer, und gegebenenfalls auch für seine Kommunikationspartner, überprüfbar macht.

Im Geschäftsverkehr agieren verschiedene Parteien mit unterschiedlichen Interes-

3 Zusätzlich zu einem Passwort wird ein Hardwareelement zur Authentifizierung verwendet. Die zwei Faktoren sind zumeist definiert als Wissen (bspw. PIN) und Besitz (bspw. EC-Karte).

4 Ein vertrauenswürdiges Rechnersystem ist ein Computersystem, welches sich immer so verhält, wie sein Benutzer es von ihm erwartet und die Daten integer und vertraulich hält.

sen und Sicherheitsstrategien: Während für Endbenutzer Datenschutzaspekte von Bedeutung sind, stellen für Unternehmen und Behörden die sichere und vertrauliche Behandlung von wichtigen Daten sowie der Schutz der Urheberrechte und Lizenzen gegen unautorisierte Verbreitung und Nutzung relevante Aspekte dar. Um die geforderte Vertrauenswürdigkeit beim Austauschen von Daten zu gewährleisten, ist es notwendig, dass die Daten mit Rechten verknüpft werden können, die auf einem fremden Rechnersystem durchsetzbar sind. Doch dürfen diese Regeln nicht mit denen des Empfängers kollidieren und trotzdem zur Ausführung gebracht werden. Ein Beispiel hierfür wäre ein Dokument, welches eine andere Person lesen darf, aber nur unter der Einschränkung, dass das Dokument weder gedruckt, noch verschickt werden kann.

Zusammengefasst werden folgende Anforderungen an eine Sicherheitsplattform definiert:

- Überprüfbarkeit der Vertrauenswürdigkeit eines Rechnersystems
- Durchsetzbarkeit von Rechten auf einem entfernten Rechnersystem im Sinne multilateraler Sicherheit
- Hardware- und Softwareunabhängigkeit
- Nutzbarkeit herkömmlicher Rechnersysteme
- Hohe Sicherheit und Vertrauenswürdigkeit

Um diese Anforderungen in Ziele münden zu lassen, wurde das EMSCB-Konsortium[5] gebildet, das gefördert durch das Bundesministerium für Wirtschaft und Technologie (BMWi) eine vertrauenswürdige, faire und offene Sicherheitsplattform realisiert (Sadeghi et al. 2004). Die aus diesem Konsortium heraus entstandenen und zukünftigen Sicherheitstechnologien und Ergebnisse werden unter dem Namen Turaya geführt. *Turaya* ist ein Fantasie-Name, der im Gegensatz zum Projektnamen *EMSCB* eingängig und leicht zu merken ist.

3 Trusted Computing mit Turaya

Trusted Computing ist eine Sicherheitstechnologie, die seit 2003 von der *Trusted Computing Group (TCG)* spezifiziert wird. Die *TCG* ist ein Industriekonsortium aus über 140 Firmen. Darunter befinden sich alle „Global Player", wie *SUN*, *Intel*, *AMD*, *Microsoft*, *HP*, *IBM* sowie *Infineon* als Promoter, aber auch weitere deutsche Hersteller wie *Fujitsu Siemens*, *Utimaco Safeware AG*, *Sirrix AG* und andere, die es sich zur Aufgabe gemacht haben, offene Spezifikationen für vertrauenswürdige Rechnersysteme zu entwickeln, um die Sicherheit verteilter Anwendungen mit vertretbarem Aufwand zu erhöhen.[6]

5 *EMSCB* steht für *European Multilaterally Secure Computing Base*.

6 Siehe die Webseite der *Trusted Computing Group* unter http://www.trustedcomputinggroup.org.

Die Hauptidee besteht darin, manipulationssichere Sicherheitskomponenten in die Hardware zu integrieren, die als vertrauenswürdige „Anker" für die Sicherheit des Rechnersystems genutzt werden sollen, indem sie die Integrität und Authentizität des Rechnersystems garantieren. Die wichtigste Komponente ist das Trusted-Platform–Modul (TPM). TPMs werden bereits eingesetzt und sind bisher vor allem in Notebooks eingebaut worden. Ende des Jahres 2006 sollten es bereits 60–100 Millionen sein (vgl. Kay 2005).

3.1 Das Trusted-Platform-Modul

Das TPM ist ein kleiner passiver Chip, der einen Mikrokontroller enthält. Der Chip ist fest mit dem Mainboard oder dem Prozessor verbunden und rein von der Architektur her mit einer *SmartCard* vergleichbar. Es beinhaltet einen Crypto-Coprozessor, einen Zufallszahlengenerator und die *Platform Configuration Register (PCR)*, in denen Hashwerte[7] von Konfigurationszuständen gespeichert werden. Um die Vertrauenswürdigkeit eines Rechnersystems zu erhöhen, bietet das TPM ein Zertifikat mit einem Schlüsselpaar, dem *Endorsement Key (EK)*, welches das TPM niemals verlässt und die Eindeutigkeit und Einzigartigkeit des TPMs definiert. Um eine gewisse Anonymität wahren zu können, werden in den meisten Anwendungsfällen stets Schlüssel verwendet, die vom *EK* abgeleitet wurden. Diese Schlüssel werden *Attestation Identity Keys (AIK)* genannt. Die *AIKs* können nur für die Signatur von Werten des PCR-Registers eingesetzt werden. Für alle weiteren Verfahren stellt der *Storage Root Key (SRK)* die Wurzel des Schlüsselbaums dar. Der Schlüsselbaum besteht aus einer nicht endlichen Menge an Schlüsseln, die mit dem *SRK* verschlüsselt werden. Diese sind für sämtliche Verschlüsselungs- und Signaturvorgänge, die von der vertrauenswürdigen Plattform kontrolliert werden, einsetzbar.

3.2 Die Funktionen

Das TPM ist ein passives Sicherheitsmodul. Um das TPM nutzen zu können, muss es zuerst vom Besitzer aktiviert werden. Dieser Vorgang wird als *take ownership* bezeichnet. Zur sinnvollen Verwendung der TPM-Funktionen wird eine Software–Sicherheitsplattform benötigt, die sämtliche Vorgänge sicher und vertrauenswürdig steuert. Herkömmliche Betriebssysteme können aufgrund der hohen Fehleranfälligkeit und der ihnen eigenen Struktur den Ansprüchen an eine solche Sicherheitsplattform nicht genügen. Es fehlen entscheidende Strukturen und Konzepte, die zum Beispiel eine strikte Trennung von Speicherbereichen ermöglichen, um bei einem Angriff den Schaden einzuschränken. Eine Authentifizierung von Applikationen oder der

7 Hashwerte sind Werte, die mit Hilfe einer Hashfunktion berechnet werden. Hier berechnet eine Hashfunktion zu einer binärwertigen Folge einen binärwertigen Block fester Länge (den sog. Hashwert; derzeit übliche Länge ist 160 Bit). Der Vergleich zweier Hashwerte erlaubt den Ausschluss von Gleichheit der Ausgangsdaten bei deren Ungleichheit. Der Hashwert entspricht einem Fingerabdruck.

Sicherheitsplattform kann bisher ebenfalls nicht gewährleistet werden, wodurch eine Anwendung nie einen nachweisbar vertrauenswürdigen Status erreichen kann. Die Sicherheitsplattform Turaya ist speziell für diesen Einsatz konzipiert worden und ist in der Lage, z. B. die im Folgenden aufgeführten und wichtigsten Sicherheitsfunktionen zu steuern.

Secure boot

Das TPM alleine bringt einem Rechnersystem noch keinen Vorteil. Es ist auch notwendig, dass vertrauenswürdige Sicherheitsmechanismen greifen, sobald das Rechnersystem startet. Die erste Instanz, die unmittelbar zu Beginn des Bootvorgangs eingreift, ist das *Core Root of Trust for Measurement (CRTM)*, welches zusammen mit dem TPM den *Trusted Building Block* bildet (vgl. Abb. 1). Der *CRTM* ist ein ausführbarer Code, der einen Messvorgang über einzelne Systemzustände durchführt und dann die Ergebnisse in den *PCR* hinterlegt.[8] An dieser Stelle wird der Grundstein für ein vertrauenswürdiges Rechnersystem gelegt. Alle folgenden Bootvorgänge bzw. Systemzustände werden kontrolliert „aufgezeichnet". Diese Aufgabe übernimmt der *trusted bootloader*. Im Falle von Turaya ist dies der *Trusted Grub*. Er lädt die folgenden Module und führt deren Messung, wie im Folgenden beschrieben, durch (vgl. Abb. 1).

Die Hardware und die Software, die zu einer vertrauenswürdigen Komponente gehören, werden mit Hilfe einer Hashfunktion gemessen. Der daraus gebildete Hashwert repräsentiert diese Messung und wird im *PCR* hinterlegt. Einige Speicherplätze des *PCR* sind für bestimmte Messungen der Hardware vordefiniert (vgl. Abb. 1). Der Zustand des Rechnersystems ist also ab diesem Moment vertrauenswürdig nachweisbar. Wird an der Konfiguration des Rechnersystems nun etwas verändert, zum Beispiel durch einen Angriff auf eine gemessene Software-Konfiguration oder durch den Austausch einer Hardwarekomponente, würde der aktuelle Konfigurationszustand nicht mehr dem gemessenen entsprechen. Wird ein solches Rechnersystem erneut gestartet, können die im vertrauenswürdigen Zustand registrierten und beispielsweise auf einem USB-Stick abgelegten Messwerte mit den Messungen des aktuellen Bootvorgangs verglichen werden. So entsteht ein Trusted-(Authenticated)-Boot-Vorgang. Stimmt ein Wert im *PCR* nicht mit dem als vertrauenswürdig geltenden Wert überein, kann dem User diese Erkenntnis beispielsweise durch eine Warnung mitgeteilt werden. Per Definition ist das System oder die Applikation in diesem Zustand nicht vertrauenswürdig. Von einem *secure boot* wird gesprochen, wenn der Bootvorgang direkt abgebrochen wird, falls das System als nicht vertrauenswürdig eingestuft wurde.

Um die beschriebenen Funktionen sinnvoll nutzen und steuern zu können, wird eine vertrauenswürdige Sicherheitsplattform benötigt, beispielsweise um Applikationen eindeutig voneinander zu trennen, damit sie separat auf Vertrauenswürdigkeit

8 Vgl. Bundesamt für Sicherheit in der Informationstechnik (2006).

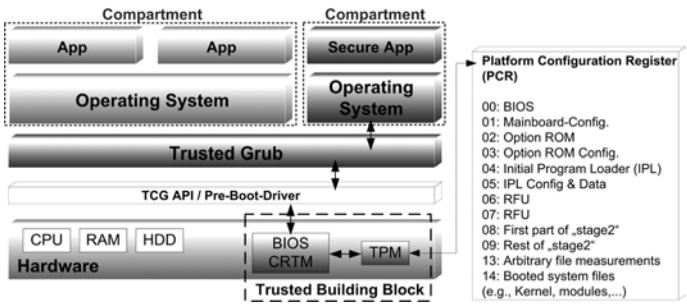


Abbildung 1: Vorgang und Architektur des secure boot unter Verwendung des Trusted Grub

überprüft werden können oder um den vertrauenswürdigen Zustand weitergeben zu können, um sich „auszuweisen". Die Messwerte lassen sich etwa nutzen, um sie mit Werten von Applikationen zu vergleichen, die vom Hersteller mitgegeben wurden und einen unveränderten Zustand widerspiegeln. Turaya ist eine solche Sicherheitsplattform. Die beschriebenen Verfahren bilden die Grundlage für alle Funktionen, die durch TC-Technologie ermöglicht werden.

Sealing (Versiegeln)

Ein Grundanliegen der Anwender ist das Bedürfnis, die eigenen Daten vor fremdem Zugriff zu schützen. Idealerweise kombiniert mit der Möglichkeit, die Daten auch an ein anderes Rechnersystem übergeben zu können mit der Gewissheit, dass die Daten auf dem entfernten Rechnersystem gemäß den eigenen Vorstellungen behandelt werden. Durch den Einsatz der TC-Technologie ist es nun möglich, Daten an eine Systemkonfiguration zu binden. Diesen Vorgang nennt man *sealing*. Durch *sealing* sind diese Daten nur auf dem Rechnersystem mit der entsprechenden Konfiguration abrufbar. Beispielsweise wird der Hashwert einer vertrauenswürdigen Konfiguration mit den zu schützenden Daten zu einem Datenpaket verbunden (vgl. Abb. 2). Hierbei kommt eine Verschlüsselung zum Einsatz, die gewährleistet, dass die Daten nur auf den Systemen wieder entschlüsselt werden können, die die entsprechende Konfiguration vorweisen. In Abbildung 2 verfügt System 1 über diese Konfiguration und kann somit das Datenpaket auslesen, System 2 jedoch nicht. Die mit einem bestimmten Konfigurationszustand verknüpften Daten könnten auch zusätzlich mit einem Schlüssel des TPMs verschlüsselt werden. Diese Daten wären dann nur auf diesem Rechnersystem mit genau diesem TPM verfügbar.

Binding (Auslagern)

Es ist zudem möglich, die Daten mit einer so genannten *security policy* für die Verarbeitung auf entfernten Rechnersystemen zu versehen. Diesen Vorgang nennt

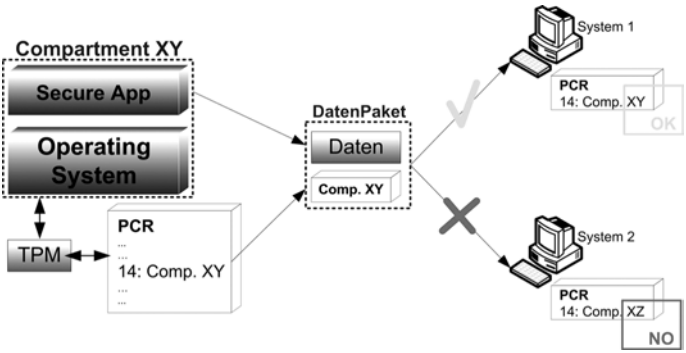


Abbildung 2: Der prinzipielle Vorgang beim sealing

man *binding*. Dadurch wird es möglich, jemand anderem seine Daten zur Verfügung zu stellen, aber nur unter bestimmten Bedingungen. Der Besitzer der Daten könnte etwa durch eine *security policy* vorschreiben, dass keine E-Mail- oder Druckertools auf dem Rechnersystem verfügbar sein dürfen, solange seine Daten genutzt werden, damit diese nicht versendet oder gedruckt werden können.

Attestation (Beglaubigung der Rechnersystemintegrität)

Zusätzlich zu dem Binden der Daten an das eigene Rechnersystem ist das genaue Wissen über den Zustand des Rechnersystems eines Kommunikationspartners ein sehr nützliches Sicherheitsfeature. Mit Hilfe der Attestierung kann der Zustand eines Rechnersystems überprüft werden. Bevor Daten an ein anderes Rechnersystem versendet werden, sollte erst sichergestellt sein, dass das andere Rechnersystem auch wirklich das Rechnersystem ist, das es vorgibt zu sein. Es sollte außerdem eine Hard- und Softwarekonfiguration aufweisen, welche die gewünschte Vertrauenswürdigkeit besitzt. Da die TPMs mit ihren Schlüsseln Einzigartigkeit mit sehr hoher Wahrscheinlichkeit gewährleisten, ist ein Rechnersystem eindeutig identifizierbar. Über eine vertrauenswürdige dritte Instanz, die die jeweiligen Rechnersysteme zertifiziert hat, wäre es also möglich, ein Rechnersystem zu identifizieren, indem sich die Rechnersysteme über die Schlüssel und Zertifikate ihrer TPMs ausweisen. Dies eröffnet erstmals die Möglichkeit, nicht nur Anwender, sondern auch die Rechnersysteme selbst authentifizieren zu können. Die Funktion wird als *(remote) attestation* bezeichnet. So ergeben sich weitere Sicherheitsmerkmale, beispielsweise für Netze mit externen Mitarbeitern.

Manipulationen durch Angreifer, die mit ihrem eigenen Rechnersystem einen Angriff durchführen, sind somit nahezu unmöglich und falls Angreifer das Rechnersystem eines externen Mitarbeiters übernehmen wollten, kämen sie weder an Daten, die durch *sealing* verschlüsselt wurden, noch könnten sie in das Firmennetz eindringen, da das gesamte Rechnersystem durch den Angriff die vertrauenswürdige Konfigurati-

on verloren hätte. In der TPM-Spezifikation 1.2 wird neben dem Einsatz einer dritten Partei auch ein Sicherheitsverfahren zur Attestierung angeboten, das ohne einen vertrauenswürdigen Dritten auskommt. Dieses Verfahren nennt sich *Direct Anonymous Attestation (DAA)*.

Diese Attestierungsfunktion besitzt Potenzial im Hinblick auf *grid computing*, Webservices, Peer-to-Peer- und Unternehmenskommunikation, da derartige gegenseitige Attestierungen das Vertrauen in solche Technologien stärken.

3.3 Wie kann Trusted Computing die Sicherheitssituation verbessern?

Der Einsatz von zusätzlicher sicherer Hardware, also beispielsweise der Einsatz von TPMs, erhöht das Maß an Sicherheit und Vertrauenswürdigkeit eines Rechnersystems beträchtlich. Durch die beschriebenen Sicherheitsfunktionen ist also eine Steigerung der Vertrauenswürdigkeit von Rechnersystemen möglich, da Daten effektiver geschützt und Anwendungen wesentlich besser kontrolliert werden können.

Es eröffnen sich die Möglichkeiten zu neuen Geschäftsfeldern, in deren Arbeitsgebiet aufgrund der Unsicherheit des Mediums *Computer* bisher keine Möglichkeit gesehen wurde, die IT zu nutzen. Firmengeheimnisse können effizient geschützt werden.

4 Die Sicherheitsplattform Turaya

Im Wesentlichen bietet die Sicherheitsplattform Turaya einen mikrokernbasierten Sicherheitskern, der „unterhalb“ von herkömmlichen Betriebssystemen agiert. Turaya ist ein eigenständiges, sicheres kleines Betriebssystem. Die gesamte Ressourcenverwaltung, die Kontrolle über Funktionen und Prozesse im Hinblick auf TC-Funktionalitäten und die Rechteverwaltung werden von Turaya übernommen. Das Konzept der Isolation durch Virtualisierung (vgl. 4.1) ermöglicht den Einsatz herkömmlicher Betriebssysteme neben hoch sicher geschützten Anwendungen. Das TPM bietet die Möglichkeit, sicherheitsrelevante Prozesse durch Hardwaresicherheit zu stützen.

4.1 Architektur

Die Architektur der Sicherheitsplattform Turaya ist so angelegt, dass sie hardware- und softwareunabhängig agieren kann. Die Architektur ist in sich abgeschlossen und bietet entsprechende Schnittstellen „nach oben“ zur Anwendungsschicht (*Application Layer*) und „nach unten“ zur Hardwareschicht (*Hardware Layer*) an.

Die Architektur des *Security Kernel* von Turaya spaltet sich in zwei Teile, den *Resource Management Layer (RML)* und den *Trusted Software Layer (TSL)*. Die Abbildung 3 zeigt die Einordnung dieser Layer in die Gesamtarchitektur eines Rechnersystems. Die dunklen Komponenten stellen ein herkömmliches, nicht vertrauenswürdiges Rechnersystem dar. Die helleren Komponenten zeigen die Sicherheitsplattform

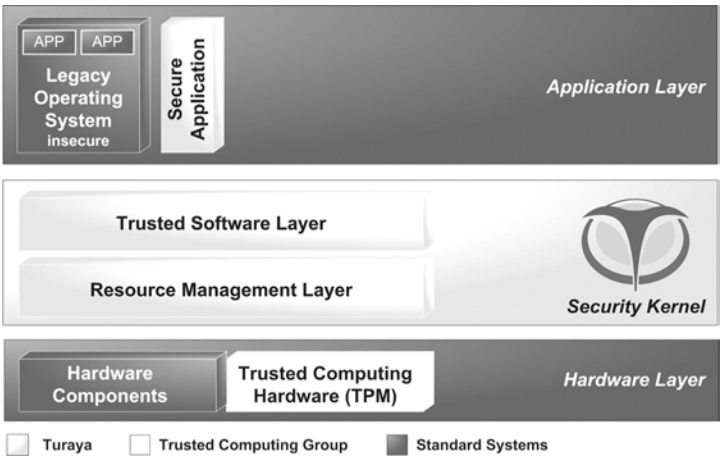


Abbildung 3: Die Turaya-Architektur im Überblick

Turaya und auf der Application-Layer-Ebene die sicheren Applikationen, die nur auf Turaya zugreifen (Linnemann und Pohlmann 2006*a*). Die hellgrau gefärbte Trusted–Computing-Hardware nimmt in diesem Zusammenhang eine besondere Stellung ein, da die Technologie lediglich durch Turaya genutzt und gesteuert wird. Ein TPM ist für den Einsatz von Turaya nicht grundsätzlich notwendig. Jedoch können die genannten TC-Funktionalitäten ohne den TPM nicht genutzt werden. Dies hätte ein wesentlich geringeres Sicherheitsniveau zur Folge.

Isolation durch Virtualisierung

Der *Resource Management Layer* Turayas steuert und kontrolliert den Zugriff auf die Hardware des Rechnersystems. Eine der wichtigsten Aufgaben dieser Schicht ist die Virtualisierung der Hardware. Umgesetzt wird dies aktuell durch einen L4-Mikrokernel[9], der zusammen mit den Gerätetreibern den *RML* bildet. Durch die Virtualisierung ist es möglich, mehrere „Rechnersysteme" zu simulieren. Diese Technik wird bereits vielfach angewendet, zum Beispiel, um auf einem realen Server mehrere virtuelle Maschinen laufen zu lassen, die jeweils wie ein selbstständiger Server agieren. Bekannte Virtualisierungssoftwaresysteme sind beispielsweise *Xen* oder *VMware*. Durch die Virtualisierung wird ein sicherheitstechnisch entscheidendes Konzept umgesetzt – die Isolation. Ein herkömmliches Betriebssystem läuft auf der Sicherheitsplattform Tura-

9 *L4* ist ein Betriebssystemkern (Kernel), der ursprünglich von Jochen Liedtke entwickelt wurde. *L4* bezeichnet heute vor allem die *API* (Anwendungsprogrammschnittstelle). In diesem Projekt wird die L4-Implementierung *Fiasco* der TU Dresden verwendet und angepasst. *L4* ist ein sehr kleiner Betriebssystemkern, der zumeist als Basis für Echtzeitbetriebssysteme verwendet wird. Sein Vorteil ist, dass er wesentlich kleiner ist als Betriebssystemkerne von herkömmlichen monolithischen Systemen.

ya, anstatt wie bisher direkt auf der Hardware. Zusätzlich ist die Sicherheitsplattform in der Lage, parallel und vollständig isoliert voneinander weitere Betriebssysteme oder Applikationen laufen zu lassen. Die strikte Trennung erlaubt es, sicherheitskritische Anwendungen auch dann auszuführen, wenn das herkömmliche, parallel laufende Betriebssystem bereits kompromittiert wurde. Das gesamte zu schützende Gut, wie beispielsweise Zertifikate, Schlüssel und *credentials* aller Art, kann auf diesem Wege vor Angriffen wirkungsvoll geschützt werden. Die einzelnen voneinander isolierten Applikationen und Betriebssysteme oberhalb der Sicherheitsplattform Turaya werden als *compartments* bezeichnet (vgl. Abb. 4).

Die Möglichkeiten der Turaya-Architektur am Beispiel Onlinebanking

Sicherheitskritische Prozesse werden durch die Virtualisierung von unsicheren Prozessen getrennt. Onlinebanking beispielsweise lässt sich mit der Sicherheitsplattform Turaya hoch sicher betreiben. Um eine entsprechende Vertrauenswürdigkeit zu erreichen, wird die Onlinebanking-Applikation in einem Compartment ausgeführt. An dieser Stelle sind zwei unterschiedliche Szenarien denkbar: Zum einen wäre es möglich, ein Compartment mit einem Betriebssystem und einem Browser auszustatten; dazu sind keine Anpassungen an der Software notwendig. Zum anderen würde eine Banking-Software so modifiziert werden, dass sie direkt auf den Schnittstellen der Sicherheitsplattform aufsetzte. Im Folgenden wird das Beispiel der modifizierten Banking-Software weiterverfolgt. Sobald der Anwender seine Banking-Applikation startet, „springt" er in das entsprechende Compartment. Für den Anwender ist dieser Vorgang transparent. Er startet die Applikation, beispielsweise durch einen Klick auf das Icon seiner Anwendung im herkömmlichen Betriebssystem. Dadurch wird ein Aufruf an die Turaya-Plattform gesandt. Diese startet die Banking-Software in einem neuen Compartment. Daraufhin öffnet sich die Anwendung in einem Fenster. Dieses Fenster ist bereits das sichere Compartment. Der Bankserver wird angewählt. Über die Attestierungsfunktion kann vertrauenswürdig nachgewiesen werden, dass der Anwender wirklich mit dem Server seiner Bank verbunden ist und sich dieser Bankserver in einem vertrauenswürdigen Zustand befindet. Die Attestierung könnte folgendermaßen realisiert werden:

Die Banking-Software besitzt beispielsweise den öffentlichen Teil eines Schlüsselpaares, der sich aus der vertrauenswürdigen Konfiguration in Verbindung mit einem *AIK* des TPMs des Bankservers zusammensetzt. Eine mit diesen Informationen verschlüsselte Zufallszahl wird an den Bankserver übermittelt. Ist dieser in der Lage, mit seinem einmaligen privaten Schlüssel die Zufallszahl zu entschlüsseln und gibt dieses Wissen an den anfragenden Server zurück, so ist die Authentizität und Vertrauenswürdigkeit des Servers sichergestellt, da nur dieser Server in der Lage sein kann, die verschlüsselte Zufallszahl zu entschlüsseln.

Der Anwender gibt seine transaktionsrelevanten Daten ausschließlich innerhalb des sicheren Compartments ein. Das herkömmliche Betriebssystem hat keinen Zu-

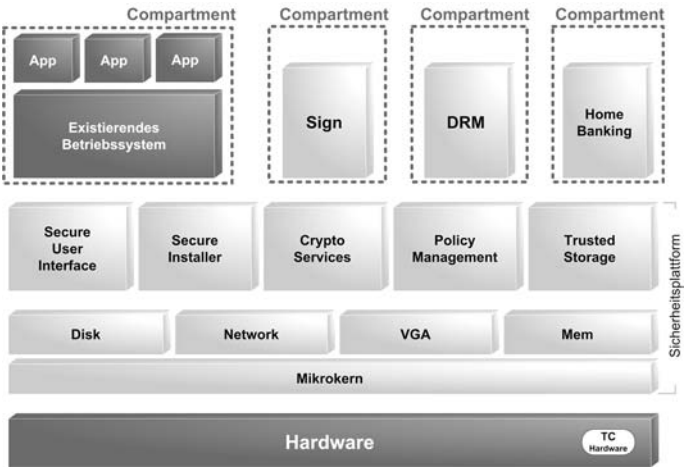


Abbildung 4: Die Turaya-Architektur im Detail

griff. Dadurch ist evtl. vorhandene Malware nicht in der Lage, die Eingaben des Anwenders zu missbrauchen. Das Compartment wurde in einem unveränderten Zustand gemessen. Wenn ein Angriff auf dieses Rechnersystem erfolgt wäre, der das Compartment beeinflusst hätte, wäre das Compartment aufgrund des beschriebenen Secure-Boot-Vorgangs jedoch erst gar nicht gestartet worden, da die Messwerte nicht mehr übereingestimmt hätten und damit die Vertrauenswürdigkeit nicht gewährleistet war.

Im Rahmen des Onlinebankings lässt sich ein weiterer Vorteil der Sicherheitsplattform Turaya lokalisieren. Das Onlinebanking-Verfahren *HBCI*[10] verlangte bisher mindestens einen teuren Klasse-3-Kartenleser für ein höheres Sicherheitsniveau. Turaya stellt eine vertrauenswürdige Middleware und somit einen *trusted path*[11] zwischen Kartenleser und Banking-Applikation dar. Dadurch ist der Einsatz eines simplen kostengünstigen Kartenlesers zusammen mit Turaya völlig ausreichend für ein hohes Sicherheitsniveau. Dieses Szenario zeigt, wie die Technologien *Trusted Computing* und *SmartCard* zusammenspielen. Das TPM ist fest mit einem Rechnersystem verbunden, während die *SmartCard* für mobile Anwendungen eingesetzt wird. So besteht die Aufgabe der *SmartCard* darin, einen Nutzer zu authentifizieren, während die Hauptaufgabe des TPMs in der Authentifizierung des Rechnersystems liegt.

10 *HBCI* steht für *Home Banking Computer Interface* – ein nationaler Standard des Zentralen Kreditausschusses.

11 Als *trusted path* wird ein Mechanismus beschrieben, der die direkte Kommunikation zwischen einer *Trusted Computing Base*, wie Turaya, und einem Terminal oder Nutzer beschreibt. Der Kommunikationsweg ist nicht manipulierbar. Er basiert darauf, dass sich alle daran beteiligten Instanzen als vertrauenswürdig ausweisen können.

Das beschriebene Beispiel zeigt die Aufgaben des *TSL*. Die Kontrolle über sämtliche Vorgänge wird in dieser Architekturschicht umgesetzt. Die sichere Installation von Komponenten, das Starten von Compartments, die Erzeugung von Zufallszahlen oder die sichere Eingabe von Nutzerdaten über die *Secure GUI*[12] sind nur einige Beispiele der Vorgänge, die in dem *TSL* ablaufen oder von diesem gesteuert werden.

Policy enforcement[13]

Zusätzlich zu diesen Sicherheitsfunktionen bietet die Sicherheitsplattform Turaya eine weitere Funktion, die innerhalb des *TSL* genutzt wird. Daten, die innerhalb der Sicherheitsplattform verarbeitet werden, können mit so genannten *policies* versehen werden. Die *policies* enthalten Regeln, die angeben, auf welche Art die Daten verarbeitet werden sollen. Turaya als vertrauenswürdige Instanz bindet die *policies* an die Daten und sorgt dafür, dass diese *policies* auf einem anderen Rechnersystem durchgesetzt werden unter der Voraussetzung, dass die *policy* mit der des Rechnersystembesitzers übereinstimmt. Eine Regel, die das Ausdrucken einer Datei untersagt, würde Turaya innerhalb des *TSL* umsetzen, indem sie dem Compartment, das diese Datei anzeigt, den Zugang zum Druckertreiber verwehren würde. Über diese Vorgänge lassen sich Digital-Rights-Management-Funktionen[14] mit der Sicherheitsplattform Turaya anwenden.

4.2 Die Applikationen

Die Sicherheitsplattform Turaya wird im Rahmen des EMSCB-Projekts entwickelt. Dieses ist ein Konsortium aus der Ruhr-Universität Bochum, dem Institut für Internet-Sicherheit der FH Gelsenkirchen, der TU Dresden sowie den Firmen *Sirrix AG* und *escrypt GmbH*. Unterstützt wird das Projekt über drei Jahre vom Bundesministerium für Wirtschaft und Technologie. Nach Ablauf dieser Zeitspanne wird Turaya im Trusted-Computing-Kompetenzzentrum, das Anfang 2007 gegründet werden soll, weiterentwickelt und gepflegt werden. Innerhalb des Projektzeitraumes sollen die folgenden fünf Piloten umgesetzt werden, die die Funktionalitäten der Sicherheitsplattform demonstrieren (Linnemann und Pohlmann 2006*b*):

1. *Turaya.Crypt* – Device-Verschlüsselung; fertiggestellt
2. *Turaya.VPN* – Sicheres VPN-Modul (Zertifikatsmanagement); fertiggestellt
3. *Turaya.FairDRM* – Faires DRM-System; fertiggestellt

12 *Secure GUI* beschreibt das vertrauenswürdige *Secure Graphical User Interface*, das vom Sicherheitskern kontrolliert wird und somit nicht von Anwendungen manipuliert werden kann.

13 Aus dem Englischen: *policy* = Verfahrensweise, Grundsatz; *enforcement* = Durchsetzung, Erzwingung.

14 *Digitales Rechtemanagement (DRM)* ermöglicht Rechteinhabern die Durchsetzung ihrer Rechte bei geschützten Daten wie etwa Musik- oder Filmdateien.

4. *Turaya.ERM – Enterprise Rights Management* (Dokumentenmanagement) mit *SAP*; November 2007

5. *Turaya (embsys)* – Embedded-Systems-Einsatz (Automotiv Multimedia); November 2007

Die ersten beiden fertig gestellten Piloten demonstrieren den Einsatz der Trusted–Computing-Technologie und die Isolation sicherheitskritischer Komponenten. Alle sicherheitsrelevanten Daten stehen in diesen Applikationen unter der Kontrolle der Sicherheitsplattform und erlauben keinerlei Zugriff durch das herkömmliche Betriebssystem. Die Piloten stehen auf der EMSCB-Webseite[15] als Demo-Image zum Testen zur Verfügung. Der dritte Pilot beschäftigt sich mit dem Einsetzen und Durchsetzen von *policies*. Auf diesen Ergebnissen aufbauend werden die Piloten vier und fünf im Verbund mit strategischen Partnern entwickelt. Das *Enterprise Rights Management* wird mit der *SAP AG* und Turaya auf Embedded-Rechnersystemen mit der Bosch/Blaupunkt-Gruppe gemeinsam umgesetzt. Dies führt zu einer marktnahen Entwicklung der Sicherheitsplattform, basierend auf Anforderungen von Herstellern.

5 Das Besondere an Turaya – TC kritisch diskutiert

Es fragt sich hier nicht nur, ob Turaya die aufgestellten Anforderungen erfüllen kann, sondern auch auf welche Weise. *Microsoft* kündigte bereits weit vor dem Jahr 2003 ein System mit dem Namen *Palladium*[16] an, das TC-Funktionalitäten unterstützt. Aufgrund der negativen Schlagzeilen um dieses Projekt änderte *Microsoft* den Namen 2003 in *NGSCB (Next Generation Secure Computing Base)*. Es wurden eine Menge von Funktionen auf Basis von *Trusted Computing* versprochen, die immer im engen Verbund zu DRM-Funktionen standen, welche im neuen Betriebssystem eingesetzt werden sollten. Im jetzt erschienenen Microsoft-Betriebssystem *Vista* ist von all den ursprünglich versprochenen Funktionen lediglich eine Festplattenverschlüsselung mit Namen *BitLocker* übrig geblieben, die das TPM nutzt.

Durch *Trusted Computing* ist es möglich, das Rechnersystem eines Anwenders zu kontrollieren und Informationen über denselben zu bekommen. Zusätzlich kann einem Anwender vorgeschrieben werden, welche Software er nutzen darf und welche nicht. Dies sind Aussagen der Gegner der TC-Technologie und die Punkte, die dazu beigetragen haben, dass die TC-Technologie in ein schlechtes Licht gerückt wurde. Tatsächlich hätte *Microsoft* die Möglichkeit, sein Betriebssystem so zu entwickeln, dass es andere Betriebssysteme oder Anwendungen von bestimmten Anbietern auf seinem Rechnersystem zur Laufzeit nicht zulässt (vgl. The Information Security Journal 2003; Dolle 2004; Caspers 2004). Dieses Szenario wird zu Recht kritisiert. Dem entgegen steht jedoch die aktuelle Entwicklung des zunehmenden Einsatzes der IT

15 http://www.emscb.de

16 Siehe http://www.microsoft.com/resources/ngscb.

im sicherheitskritischen Umfeld. Hier ist eine Sicherheitsplattform, wie einleitend beschrieben, für innovative und hochsichere Anwendungen notwendig.

Die Sicherheitsplattform Turaya wählt daher einen eigenen Ansatz, der die Vorteile der TC-Technologie nutzt und die Diskriminierung von Anwendern oder Anbietern verhindert. Turaya ist eine Open-Source-Sicherheitsplattform, die unter der GPL–Lizenz veröffentlicht wird. Dies ermöglicht dem Anwender genau zu verfolgen, ob die Sicherheitsplattform schon von der Struktur und Funktionsweise her in irgendeiner Form die Rechte des Anwenders untergräbt. Hintertüren, wie sie in Closed-Source–Systemen vermutet und eingesetzt werden könnten, wären in der Sicherheitsplattform Turaya sofort sichtbar. Bei der Konnektivität setzt Turaya auf offene Standards und möchte in diesem Bereich selbst einen offenen Standard definieren. Zur Steigerung der Vertrauenswürdigkeit und um den Einsatz im Hochsicherheitsbereich zu ermöglichen, wird eine Evaluierung z. B. nach *Common Criteria EAL 4*[17] angestrebt.

Um die Verarbeitung von Daten und Dokumenten auf fremden Rechnersystemen vertrauenswürdig durchführen zu können und um Datenschutz und Urheberrecht zu gewährleisten, ist der Einsatz von *policies* notwendig. In diesem Zusammenhang lassen sich, wie bereits erwähnt, auch DRM-Funktionalitäten einsetzen. Turaya wählt hierfür einen fairen Ansatz: Ein Anwender definiert seine eigenen *policies*. Wenn eine Datei oder Anwendung eines Anbieters auf dem Rechnersystem ausgeführt werden soll, die als *policy* vorgibt, dass die Datei nur einmal angezeigt werden kann und dafür ein bestimmter Geldwert zu entrichten ist, wird dies nur durchgesetzt, wenn die *policy* des Anwenders dies zulässt. Diese Eigenschaft wird als multilaterale Sicherheit bezeichnet. Somit ist es selbstverständlich möglich, Rechte auf fremden Rechnersystemen durchzusetzen, aber nur, wenn der Anwender des Rechnersystems dies zulässt.

Hinzu kommt, dass die Sicherheitsplattform Turaya sowohl hardware- als auch softwareunabhängig ist. So gibt es einerseits keine Diskriminierung von Anbietern und andererseits ein sehr hohes Maß an Kompatibilität.

6 Fazit

Der Bedarf für vertrauenswürdige Rechnersysteme in verschiedensten Geschäftsfeldern und Anwendungen scheint klar ersichtlich und notwendig. In diesem Kontext sollten Daten- und Urheberschutzaspekte berücksichtigt werden. Eine Sicherheitsplattform, die den Anforderungen eines vertrauenswürdigen Rechnersystems entspricht, darf weder Anwender noch Anbieter benachteiligen.

Die vertrauenswürdige, faire und offene Sicherheitsplattform Turaya berücksichtigt diese Aspekte und bietet die notwendige Sicherheit und Vertrauenswürdigkeit für Rechnersysteme. Die definierten Anforderungen können vollständig von Turaya erfüllt werden, indem die Trusted-Computing-Technologie im positiven Sinne genutzt

17 *Common Criteria* ist ein ISO-Standard für die Bewertung der Sicherheit von Informationstechnologien.

wird. Sämtliche Programmierschnittstellen von Turaya und der Sourcecode aller sicherheitskritischen Komponenten werden zu Evaluierungszwecken offengelegt, um die Vertrauenswürdigkeit der Implementierung zu erhöhen. Turaya ermöglicht daher auch der Open-Source-Gemeinde, „konkurrenzfähig" zu bleiben.

Zudem bietet Turaya den Vorteil, dass alle sicherheitskritischen Komponenten und Anwendungen unabhängig von „klassischen" Betriebssystemen agieren können und damit für zukünftige plattformübergreifende verteilte Anwendungen optimal geeignet sind. Die Autoren wünschen sich, dass möglichst viele Anwendungen auf die Sicherheitsplattform Turaya aufbauen, um eine sichere und vertrauenswürdige Zukunft von Rechnersystemen zu ermöglichen.

Literatur

Barnitzke, A. (2005), 'Exploit und Wurm folgen dem Patch schon auf dem Fuße', *Computer Zeitung* **35**, S. 6.

Bundesamt für Sicherheit in der Informationstechnik (2006), 'Das Trusted Platform Module', http://www.bsi.de/sichere_plattformen/trustcomp/infos/tpm_report/tpm_crtm.htm [20. Dez 2006].

Caspers, T. (2004), 'Der schmale Grad zwischen Vertrauensbeweisen und Datenschutz', <*kes*> – *The Information Security Journal* (6), S. 35.

Dolle, W. (2004), 'Trusted Computing: Stand der Dinge', <*kes*> – *The Information Security Journal* (4), S. 20.

Kay, R. L. (2005), 'The Future of Trusted Computing'. https://www.trustedcomputinggroup.org/home/IDC_Presentation.pdf [20. Dez 2006].

Linnemann, M. und Pohlmann, N. (2006*a*), Die vertrauenswürdige Sicherheitsplattform Turaya, *in* P. Horster (Hrsg.), 'DACH Security 2006', syssec Verlag.

Linnemann, M. und Pohlmann, N. (2006*b*), 'Schöne neue Welt?! Die vertrauenswürdige Sicherheitsplattform Turaya', *IT-Sicherheit – Management und Praxis* (3).

Sadeghi, A., Stüble, C. und Pohlmann, N. (2004), 'European Multilateral Secure Computing Base – Open Trusted Computing for You and Me', *Datenschutz und Datensicherheit* (9), S. 548–554.

The Information Security Journal (2003), 'Vertrauenskrise – Einsichten und Aussagen vom Trusted-Computing-Symposium', <*kes*> – *The Information Security Journal* (4), S. 12. http://www.kes.info/archiv/online/03-4-012.htm [25. Jan 2007].

Windows IT Pro (2004), 'Die Bedrohung von morgen', *Windows IT Pro* (4), S. 14.

NetBSD – Das portabelste Betriebssystem der Welt*

HUBERT FEYRER, STEFAN SCHUMACHER UND MARK WEINEM

(CC-Lizenz siehe Seite 563)

NetBSD ist ein modernes, vollständig freies Betriebssystem, das kontinuierlich weiterentwickelt wird. Es wird wie viele interessante Open-Source-Projekte leider wenig beachtet. Dabei eignet sich das NetBSD-Betriebssystem für jeden Anwendungszweck und es gibt kaum einen Computer, auf dem es nicht genutzt werden kann. Wegen seiner Qualität und liberalen Lizenz wird es ebenso gerne in der Lehre wie bei der Entwicklung neuer kommerzieller Produkte eingesetzt. Dieser Artikel beschreibt die wichtigsten Merkmale von NetBSD sowie dessen Einsatzmöglichkeiten im privaten und im kommerziellen Umfeld. Eine Betrachtung des pkgsrc-Paketsystems, das auch unabhängig von NetBSD eingesetzt werden kann, schließt den Artikel ab.

Schlüsselwörter: Betriebssystem · BSD · Distribution · NetBSD

1 Was ist NetBSD?

NetBSD[1] wurde 1993 als Nachfolgeprojekt des von der *University of California, Berkeley* herausgegebenen UNIX-ähnlichen Betriebssystems *Berkeley System Distribution (BSD)* gestartet. Es ist ein vollständiges, UNIX-ähnliches Betriebssystem, das den aktuellen Open-Source- und Security-Kriterien entspricht. Gleichzeitig werden Industriestandard-APIs und Vernetzungsprotokolle sowie eine große Anzahl an Hardware-Plattformen unterstützt. NetBSD eignet sich dabei für alle Anwendungsgebiete: robuste Server-Systeme und Workstations ebenso wie Handheld-Rechner und Embedded-Systeme.

Seit seiner Entstehung ist NetBSD an der Spitze der Entwicklung von Open–Source-Betriebssystemen anzutreffen. In vielen Fällen diente NetBSD als vollständige Grundlage oder Referenz für andere Projekte.

* Enthält Text aus: NetBSD „allgemeiner“ und Security Flyer, von Stefan Schumacher, 08. Jan. 2006; sowie clt6-pkgsrc.tex von Hubert Feyrer.

1 Siehe auch http://www.netbsd.org.

NetBSD wird oft gewählt, um neu entwickelte Hardware anzusteuern, u. a. in den Bereichen Netzwerk-Computer, Ein-Platinen-Rechner und selbst Webcams und Roboter. Weiterhin wird es weltweit im Bereich der Netzwerkentwicklung eingesetzt. Internet-Service-Provider benutzen NetBSD wegen seines breiten Spektrums an Netzwerkmöglichkeiten, und begeisterte Liebhaber entscheiden sich für NetBSD aufgrund seiner breiten Hardware-Unterstützung.

Das System läuft auf einer großen Palette von Hardware-Plattformen, angefangen von der *VAX 11/750* bis zu modernsten PCs und Windows-CE-Palmtops. Egal, ob man den alten Computer vom Dachboden entstauben oder einen neuen Opteron-Server in Betrieb nehmen will: NetBSD ist bereit dafür. Durch das einfache Entfernen optionaler Komponenten eignet sich NetBSD auch sehr gut für Embedded-Systeme. Zurzeit unterstützt NetBSD 55 verschiedene Hardware-Plattformen auf insgesamt 17 CPU-Architekturen von der *DEC Alpha* über *Sparc64* bis hin zum *ARM* – alle aus demselben Quellcode.

Die am NetBSD-Projekt beteiligten Entwickler legen hohen Wert auf sauberen Quellcode, ausgereifte Lösungen, Sicherheit und Portabilität. Aufgrund dieser Philosophie bietet NetBSD ausgezeichnet les- und wartbaren Code, der die Grundlage für viele weitere Projekte bildet. Um die legendäre Portabilität des Systems auf mehr als fünfzig verschiedene Plattformen zu gewährleisten, wird strikt zwischen maschinenabhängigem und maschinenunabhängigem Quellcode getrennt. Dies führt neben der bekannten und hochgeschätzten Stabilität auch zu einem exzellent strukturierten Systemkern (Kernel), der in einer einzigen Konfigurationsdatei angepasst werden kann. Weiterhin hält sich das Projekt an bestehende Dokumentationen und Standards, wie beispielsweise *POSIX*[2], und präsentiert sich so als ein hervorragendes System zur Forschung und Lehre im akademischen Umfeld.

Das Cross-Kompilieren von Kernel, *Userland* und dem *X Window System* als grafischem Unterbau wird von den Standardwerkzeugen unterstützt. Damit kann man das komplette System für ältere, langsamere aber auch Industrie- und Embedded-Rechner auf einem schnelleren Rechner einer beliebigen Architektur kompilieren.

NetBSD bietet ein überschaubares System, in dem Kernel und *Userland* integriert sind. Weitere Software ist durch das NetBSD-Paketsystem einfach installierbar. Durch die Trennung von Anwendungen und Basissystem können Updates von Anwendungen nichts am Basissystem ändern. Der vollständige Quellcode ist, inklusive Entwicklungsgeschichte, via anonymem *CVS*[3], *CVSweb*[4], *SUP*[5] und *rsync*[6] verfügbar.

2 Aus dem IEEE Standard 1003.1-1988 herausgehend wird hier definiert, wie sich ein UNIX(-ähnliches)-System verhalten soll. Der Standard wird in der aktuellsten Version von der *Open Group* als wesentlicher Bestandteil der *Single UNIX Specification* herausgegeben und ist frei erhältlich. Siehe dazu auch http://www.opengroup.org/austin/papers/posix_faq.html.

3 Checkout via anoncvs@anoncvs.netbsd.org:/cvsroot.

4 Siehe http://cvsweb.netbsd.org.

5 *SUP* steht für *Software Update Protocol*.

6 Siehe http://rsync.samba.org.

Außerdem erstellt das NetBSD-Projekt zu jedem neuen Release Images für Installations-CDs, die von weltweit verteilten FTP-Spiegeln[7] bezogen werden können. Mehrere Anbieter vertreiben NetBSD auf CD und DVD.[8]

Die Unterstützung der unterschiedlichsten Netzwerktechnologien wie *WLAN IEEE 802.11*, *Gigabit-* und *10-Gigabit-Ethernet*, *ATM*, *Bluetooth*, *HIPPI*, *FDDI*, *HSSI*, *ARCnet* und *Token-Ring* ist bereits im Basissystem gegeben. NetBSD war zudem das erste Open-Source-Betriebssystem, das *USB*, *USB2* und *PCMCIA Audio* unterstützte.

Zusätzlich zum System bietet NetBSD eine Vielzahl ausgereifter Dokumentationen. Neben einem umfangreichen Benutzerhandbuch, das viele Aspekte der Administration beschreibt, gibt es mehrere Artikel und Bücher, die für Entwickler den Aufbau des Systems im Detail beschreiben. Standardmäßig kommt NetBSD mit einer umfangreichen Sammlung an *man pages* zu allen Programmen, Bibliotheken, Funktionen und sonstigen Themenfeldern des Systems. Weiterhin existieren zahlreiche Informationskanäle und Hilfsquellen im gesamten Internet, von Mailinglisten über Newsgroups bis hin zu Webforen und Wikis.

Auch außerhalb des Internets ist NetBSD auf Messen und Veranstaltungen präsent: Entwickler und Anwender präsentieren das System sowohl in Vorträgen als auch live, Interessierte werden beraten und bei offenen Fragen und Problemen unterstützt.[9]

2 Projektstruktur

BSD unterstützte zum Zeitpunkt seiner Entstehung bereits eine breite Palette an unterschiedlichster Hardware, und so lag das Vorhaben nahe, NetBSD als System zu erstellen, das diese verschiedenen Hardware-Architekturen unterstützt und weiter ausbaut. Hier grenzt sich auch NetBSD vom zweiten BSD-Nachfolger *FreeBSD* ab, dessen Fokus auf Optimierung für die damals an Verbreitung gewinnende PC-Plattform liegt. Zur Namensgebung bei NetBSD hat das damals ebenfalls heranwachsende Internet beigetragen – Grundidee war es, dass sich alle Interessierten über das „Netz" (engl. *net*) koordinieren und zusammenarbeiten.

Die Koordination des Projekts sowie die Integration der Arbeiten aus Forschungseinrichtungen, Universitäten und Firmen wurden vom Core-Team überwacht. Mit der Zeit wuchs die Anzahl der Beiträge, und es wurde klar, dass nicht mehr nur das Core-Team Schreibzugriff auf den NetBSD-Quellcode brauchte, sondern auch weitere Entwickler mit Schreibrechten auf den Quellcode ausgestattet werden mussten. So konnte das Core-Team entlastet und der Entwicklungsprozess vereinfacht werden. Heute arbeiten weltweit mehr als 300 Entwickler für das NetBSD-Projekt.

7 Siehe http://ftp.netbsd.org/pub/NetBSD/.

8 Siehe http://www.netbsd.org/Sites/cdroms.html.

9 Eine Übersicht der Veranstaltungen, auf denen NetBSD präsentiert wird, findet sich unter http://www.netbsd.org/gallery/events.html.

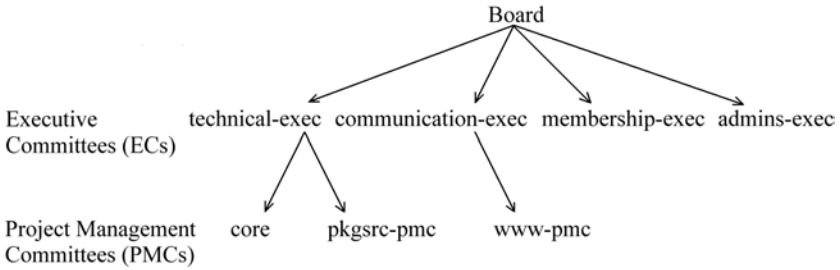


Abbildung 1: Organisation der NetBSD Foundation

Mit der Anzahl der Entwickler wuchs auch der IT-Aufwand: Es reichte nicht mehr aus, dass ein einzelner Rechner für Entwicklung (*CVS*), Bereitstellung (*FTP*), Fehlerdatenbank (*GNATS*) und Mailinglisten zuständig war, und bei der Verteilung der Aufgaben auf mehrere Rechner wollten weitere Aspekte berücksichtigt werden, wie Absicherung gegen unberechtigte Zugriffe, Datensicherheit durch Backup, sowie Wartung des Betriebssystems durch möglichst gleiche Hard- und Software.

Es wurde schnell klar, dass weitere Verwaltungsstrukturen nötig waren, u. a. ein ganzes Team von Administratoren, das die projektinternen Rechner betreut. Weitere Verbindlichkeiten entstanden, oft wurden und werden hier rechtsverbindliche Ansprechpartner benötigt. Dies galt auch für die Annahme von Spenden. Als Projekt von Freiwilligen hat NetBSD keine Firma im Hintergrund, die sich um Rechner, Administration, Anbindung und nicht zuletzt Finanzierung kümmert.

Aus diesem Bedarf an Formalisierung wurde die *NetBSD Foundation* gegründet. Ziel war und ist es, einerseits rechtsverbindlich für das NetBSD-Projekt Verträge für Hardware, Colocation und Netzwerk-Connectivity eingehen zu können, aber auch als Eigentümer der Werte des NetBSD-Projektes auftreten zu können: Hierunter fallen neben der Hardware der diversen Server v. a. das geistige Eigentum wie das Urheberrecht am NetBSD-Betriebssystem und *pkgsrc* sowie die zugehörigen Warenzeichen. Zudem kann die *NetBSD Foundation* in den USA als gemeinnützig anerkannter Verein Spenden annehmen und entsprechende Quittungen ausstellen, was zu einer einfacheren Unterstützung des NetBSD-Projekts durch Firmen führt.

An der Organisationsstruktur der *Internet Engineering Task Force (IETF)* orientiert, geschah dann auch die Aufteilung der *NetBSD Foundation* in mehrere sog. *Executive Committees (ECs)*, welche die Arbeit von mehreren *Project Management Committees (PMCs)* sowie letztendlich der in diesen Gruppen arbeitenden NetBSD–Entwickler überwachen (siehe Abbildung 1). Dies alles geschieht unter der Oberaufsicht des *Board of Directors*, dem Vorstand.

Die technische Entwicklung wird von *technical-exec* überwacht, wobei hier zwei Teilbereiche existieren: Die Entwicklung des NetBSD-Betriebssystems wird wie gehabt vom Core-Team geleitet, über die Entwicklung von *pkgsrc* als systemüber-

greifende Paketsammlung wacht das *pkgsrc-PMC*. Die Öffentlichkeitsarbeit wird von *communication-exec* überwacht, wobei hier die Wartung des Webservers an das *www–PMC* ausgegliedert ist.

Membership-exec ist die Personalverwaltung des NetBSD-Projektes und rekrutiert – üblicherweise in Zusammenarbeit mit weiteren Entwicklern – neue Entwickler und kümmert sich um die nötigen Unterlagen, um die Mitgliedschaft der neuen Entwickler in der *NetBSD Foundation* in Übereinstimmung mit der Satzung[10] zu sichern. Der *finance-exec EC* ist die Abteilung, die sich um den Finanzfluss des NetBSD–Projekts kümmert: Annahme von Spenden, Ausstellung von Spendenquittungen, aber auch Verwaltung der Konten des Projektes sowie finanzielle Abwicklung von Käufen z. B. für neue Rechner oder Ersatz-Hardware, wenn 'mal wieder eine Festplatte kaputt ist oder zu klein wird.

Die letzte, jedoch nicht unwichtigste Gruppe bildet *admins-exec*, das sich um die Betreuung der Rechner des NetBSD-Systems kümmert, Software und Betriebssysteme installiert und aktualisiert, Datensicherung betreibt und auch über die Sicherheit der Rechner wacht.

Neben den eingetragenen Entwicklern des NetBSD-Projekts tritt die zweite wichtige Gruppe in Abbildung 1 nicht explizit auf: Anwender, Administratoren, Programmierer und Entwickler. Obwohl für sie keine Mitgliedschaft in der *NetBSD Foundation* erforderlich ist, tragen sie doch entscheidend zur Weiterentwicklung von NetBSD bei: durch Erfahrungs- und Fehlerberichte, Einsenden von Erweiterungen für neue Features, Treiber oder neue Subsysteme, Dokumentation sowie auch durch ihre Entscheidung für NetBSD beim Entwickeln neuer Produkte und Dienste und nicht zuletzt durch finanzielle Unterstützung.

Wie bei allen Open-Source-Projekten spielt auch bei NetBSD neben der Projektleitung die Community die wichtigste Rolle für die Zukunft des Projektes.

3 NetBSD in der Geschäftswelt

3.1 Die BSD-Lizenz

Die Stärken von Open-Source-Betriebssystemen gegenüber herkömmlichen kommerziellen Betriebssystemen werden von immer mehr Firmen erkannt. Beim Wunsch, Produkte auf Basis der existierenden Open-Source-Systeme anzubieten spielt deren Lizenz eine wichtige Rolle. Während z. B. die *GNU General Public License (GPL)* explizit fordert, dass Änderungen an GPL-Code wieder der Öffentlichkeit zugängig gemacht werden müssen, ist dies bei NetBSD nicht der Fall, da hier die BSD-Lizenz benutzt wird. Diese erlaubt es, NetBSD als Basis für kommerzielle Systeme zu verwenden und eigene Anpassungen zum Schutz von Wettbewerbsvorteilen für sich zu behalten.

10 Siehe http://www.netbsd.org/Foundation/bylaws.html.

3.2 Offenlegung zur Wartung

Obwohl Herstellern damit die Freiheit eingeräumt wird, ihre Änderungen am Code für sich zu behalten, so spricht andererseits doch eine Reihe von Gründen dafür, Änderungen trotzdem wieder an NetBSD zurückzugeben:[11] Zum einen können künftige Entwickler und Projekte auf ihren gemachten Erfahrungen aufbauen. Zum Anderen kann dadurch ein Peer-Review für den Quellcode entstehen, was vor allem bei sicherheitskritischen Bereichen besonders relevant ist. Zuletzt spricht der Wartungsaspekt für die Rückführung von Änderungen: Da NetBSD permanent weiterentwickelt wird, müssen sonst die Hersteller die am NetBSD-Code durchgeführten Änderungen selbst nachvollziehen, um ihre Software zu integrieren. Bei einer Rückführung des Codes an das NetBSD-Projekt werden diese Änderungen im Rahmen der Entwicklungsarbeiten an NetBSD eingepflegt.

3.3 Anwendungsbeispiele

Hier einige Anwendungen, bei denen NetBSD als Basis fungierte:

- Teile des Systems wurden in andere Betriebssysteme integriert, z. B. *Apples Darwin*[12], *Castle Technology Ltds RiscOS*[13], *FMSLabs' RTCore/BSD*[14], *QNX*[15].
- Diverse WLAN-Router von *Allied Telesys*[16], *IIJ/Root Inc*[17].
- *Avocent* KVM Switches[18].
- *Brocade Rhapsody* SAN Switches[19].
- *TeamASAs Npwr*[20]: *NAS* Server mit *iSCSI*, *FiberChannel*, *Xscale*.
- *Seclaritys SiNic*[21]: *Wireless router on a card* für den Militäreinsatz.
- *Ricoh Co. IPSiO*[22]: Kopierer, Drucker, FAX, Scanner.

11 Siehe *Corporate reasons for BSD over GPL* unter http://www.feyrer.de/NetBSD/blog.html/nb_20050209_2138.html.
12 http://www.apple.com/macosx/
13 http://www.drobe.co.uk/riscos/artifact1111.html
14 http://www.fsmlabs.com
15 http://www.qnx.com/licensing/published/eula/TPOSLTG1_01.html
16 http://www.allied-telesis.co.jp/products/list/wireless/wr54id/catalog.html
17 http://www.iij.ad.jp/en/pressrelease/2005/0406.html
18 http://www.avocent.com/web/en.nsf/AttachmentsByTitle/590-356-001D(SVIP).pdf/%24FILE/590-356-001D(SVIP).pdf
19 http://www.byteandswitch.com/document.asp?doc_id=29068
20 http://www.teamasa.com/npwr_netbsd.shtml
21 http://www.seclarity.com/products/wireless/
22 http://www.ricoh.co.jp/imagio/mono/

– Überwachungskameras und Webcams von *SGI*[23], *Panasonic*[24] und *Brains Inc.*[25].
– Zahllose Evaluation-Boards, SBCs und Spezialcomputer von *Broadcom*[26], *Chalice Tech*[27], *Digital/Compaq/HP*, *NEC*[28], *BMF*[29].
– *Speecys*[30]: Humanoider Roboter mit NetBSD & PowerPC.
– *Sonys* Network Stack der *Playstation Portable* und der Emotion-Engine–EEnet-Bibliothek basiert auf dem TCP/IP Stack von NetBSD[31].

Änderungen, Patches und Quellcode von externen Entwicklern werden über Mailinglisten, Problemberichte und persönliche Kontakte akzeptiert. Bei hinreichender Qualität des Quellcodes werden den Entwicklern Schreibrechte erteilt, wobei jedoch stets klar die Interessen des NetBSD-Projekts im Vordergrund stehen, auch bei NetBSD-Entwicklern, die von Firmen bezahlt werden.

4 NetBSD – Sicherheit frei Haus

Sicherheit sollte ein integraler Bestandteil moderner Betriebssysteme und als solcher kein Zusatzfeature, sondern eine Selbstverständlichkeit sein. Neben der Untersuchung, dem Dokumentieren und dem Verbessern von Code bezüglich aktueller Sicherheitslücken beschäftigen sich die NetBSD-Entwickler auch mit regelmäßigen Inspektionen des Codes, um potenzielle Sicherheitslücken zu finden und auszubessern. Fragen zur Sicherheit werden bei NetBSD vom *NetBSD Security Officer* und dem *NetBSD Security Alert Team* behandelt.

NetBSD hat *Kerberos 5 (Heimdal)*, *OpenSSL*, *OpenSSH* und *IPsec* für *IPv4* und *IPv6* in das System integriert. Zusätzlich sind alle Dienste prinzipiell nach der Installation abgeschaltet. Neben verschiedenen Programmen zur Sicherheit bietet NetBSD durch die saubere Implementierung kaum Angriffspunkte. Außerdem ist die Administration eines NetBSD-Systems aufgrund der strikten Trennung von Basissystem und Anwendungsprogrammen wesentlich einfacher und übersichtlicher als bei vielen anderen Betriebssystemen. All das zahlt sich aus: In den anerkannten Foren zum Thema „Sicherheit" überzeugt NetBSD konstant durch sehr wenige Sicherheitsprobleme.

23 http://www.sgi.co.jp/solutions/security/pdfs/viewranger_english.pdf
24 http://panasonic.co.jp/pcc/products/en/netwkcam/lineup/c10a.html
25 http://www.brains.co.jp, http://www.netbsd.org/Changes/1999.html#mmeye
26 http://www.broadcom.com/products/1250.html
27 http://www.simtec.co.uk/products/EB110ATX/intro.html
28 http://www.engadget.com/entry/1234000293053108/
29 http://www.bm-f.com/products/overview.html
30 http://www.speecys.com
31 http://www.feyrer.de/NetBSD/blog.html/nb_20060621_2109.html

Explizit sind in NetBSD verschiedene Sicherheitsmechanismen integriert die sicherstellen sollen, dass unerwünschte Änderungen und Eingriffe am System verhindert oder zumindest erkannt werden:

Security Advisories – Sicherheitshinweise: Wenn ernsthafte Sicherheitsprobleme in NetBSD gefunden und behoben werden, wird ein *Security Advisory* veröffentlicht, das das Problem beschreibt und einen Verweis auf die Lösung enthält. Diese Anweisungen werden neben der Veröffentlichung auch auf der Projektseite archiviert.

File flags und Kernel Security Level: *File flags* ermöglichen es den Benutzern oder dem Administrator, Dateien mit bestimmten *flags* zu markieren und so vor Manipulationen zu schützen. *Kernel Security Levels* schränken bestimmte Systemfunktionen ein und ermöglichen den Einsatz von *file flags*. Beispiel: Ist der *Security Level 2* aktiviert, so sind alle Datenträger nur lesbar verfügbar und können nicht mehr ein- oder ausgehängt werden. Der TCP/IP-Filter kann nicht mehr verändert werden, und die Systemzeit lässt sich nur noch vor-, aber nicht zurückstellen.

Dateimanipulationen erkennen: `mtree` ist ein Instrument, um den Ist-Stand eines Dateisystems mit einem Soll-Stand zu vergleichen. Eingesetzt wird es vor allem, um installierte Binärdateien gegen eine vorher spezifizierte Liste abzugleichen und um Manipulationen an Dateien aufzudecken. Dazu erstellt man einen Fingerabdruck eines Dateisystems, in dem Informationen zu den Dateien abgelegt werden. Dieser Fingerabdruck kann dann gegen das laufende System abgeglichen werden und hilft so zuverlässig beim Aufdecken von Veränderungen an Dateien (beispielsweise durch Würmer, *Rootkits* oder Ähnliches).

Trojaner aussperren mit veriexec: Der NetBSD-Kernel unterstützt mit `veriexec` ein System um, das Ausführen von manipulierten Binärdateien zu verhindern. Hierzu wird die Prüfsumme eines Programms in den Kernel geladen und beim Start des Programmes mit der aktuellen Prüfsumme verglichen. Wurde das Programm z. B. von einem Wurm, *Rootkit* oder Einbrecher verändert, so verweigert der Kernel die Ausführung.

Partitionen verschlüsseln mit cgd: Mit *cgd* kann man beliebige Partitionen (außer „/“ selbst) auf Blockebene verschlüsseln. Ohne Angabe des Passworts hat niemand, auch nicht der Inhaber eines Administratorpasswortes oder jemand mit physikalischem Zugriff auf die Festplatte, eine Chance, an die Daten heranzukommen. Mit *cgd* kann man auch die Swap- und Temp-Partitionen verschlüsseln, um zu verhindern, dass dort gespeicherte geheime Daten öffentlich werden. Nach „oben“ verhält sich *cgd* wie eine Festplatte und kann mit jedem beliebigen Dateisystem versehen werden.

System calls kontrollieren mit systrace: Niels Provos' *systrace* erlaubt es, einzelne Systemaufrufe eines Programms zu kontrollieren. So ist es beispielsweise möglich, den *Apache* von einem normalen Benutzerkonto aus zu starten, weil dieser Benutzer via *systrace Apache* an den Port 80 binden darf – so dass *Apache* selbst nicht mehr mit Root-Rechten läuft.

Tägliche Sicherheitsüberprüfung: Die beiden Shellskripte „/etc/daily" und „/etc/security" erlauben es, das gesamte System auf Sicherheitslücken hin zu untersuchen. Sie können zu bestimmten Zeiten von *cron* gestartet werden und generieren einen umfassenden Bericht zu Sicherheitsproblemen.

Software auf Sicherheitslücken prüfen: Durch das Paket *audit-packages* können die installierten Pakete mit einer vom NetBSD-Projekt gepflegten Liste von bekannten Sicherheitslücken abgeglichen werden. So werden alle Pakete mit Sicherheitsproblemen aufgelistet und können dementsprechend aktualisiert werden.

Paketfilter und Firewalling: Mit *IPFilter* und *PF* unterstützt NetBSD im Basissystem zwei ausgereifte Paketfilter, die jedes NetBSD-System zum Betrieb einer ausgereiften und stabilen Firewall befähigen.

Umfangreiche Sicherheitspakete: Mit *pkgsrc* lassen sich viele weitere wichtige Sicherheitspakete installieren. Dazu zählen beispielsweise *snort*, *AIDE*, *Tripwire*, *CFS*, *chkrootkit*, *Nessus*, *Amap*, *GnuPG* und *honeyd*.

5 Aktuelle Applikationen mit pkgsrc

5.1 Herausforderungen bei der Softwareinstallation

Die Installation von Open-Source-Software unter *UNIX* ist generell mit einigen Problemen behaftet: Zum einen gibt es in der Regel eine Fülle von unterschiedlichen Softwarelösungen, die darüber hinaus häufigen Versionswechseln unterworfen sind. Zum anderen ist das Kompilieren von Quellcode teilweise zeitaufwändig und fehlerbehaftet. Zusätzlich kann auch die mangelnde Portabilität einer Software zu Problemen führen.

Hinzu kommt, dass Software-Installationen aus dem Quellcode heraus selbst für Experten oft eine Herausforderung sind: Der Anwender muss über ein solides Grundwissen über die benötigten Werkzeuge verfügen, außerdem existieren meist verschiedene Arten von Konfigurationen, die alle ihre jeweiligen Eigenheiten haben. Hinzu kommen die Abhängigkeiten von anderen Programmpaketen. Sollte dann einmal ein Fehler bei der Installation auftreten, wird zu dessen Behebung oft nicht vorhandenes Expertenwissen benötigt.

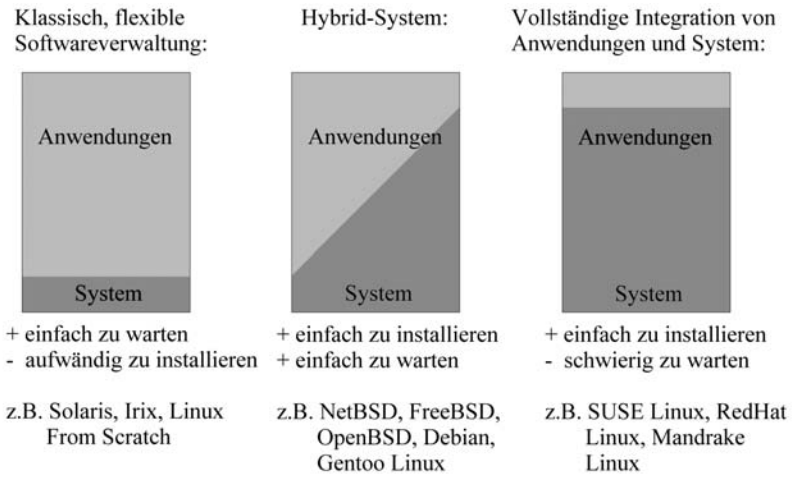


Abbildung 2: Aufwand verschiedener Konzepte für die Installation und Wartung von Betriebssystemen

5.2 Lösungen: Je nach Umgebung!

Die Lösung der im vorigen Abschnitt geschilderten Probleme kann je nach Anforderungen und Umgebung anders aussehen: Falls sich Software wenig ändert (z. B. bei vorgefertigten Binärdistributionen), empfehlen sich automatisierte Installationsroutinen. Dieser Ansatz wird vor allem bei Desktop-Systemen wie *Windows* oder *SUSE Linux* verwendet, wobei einfachere Installation mit komplexerer Wartbarkeit erkauft wird. Steht der Wunsch nach einfacher Wartung im Vordergrund (z. B. bei Webservern mit *Solaris*, *Apache* und *PHP*) empfiehlt es sich, ein stabiles Grund–Betriebssystem und wichtige Pakete selbst zu kompilieren. Das Ergebnis ist ein leicht wartbares System, welches jedoch mehr Arbeit beim Einrichten erfordert. Eine ideale Lösung aus beiden Welten stellen Systeme dar, die sowohl eine einfache Installation als auch flexible Wartbarkeit bieten. Ein Beispiel für eine derartige Lösung ist das von NetBSD verwendete *pkgsrc*[32].

Mit Hilfe des Paketsystems *pkgsrc* können mittlerweile über 6 000 frei erhältliche Software-Pakete (z. B. *KDE*, *GNOME*, *XFCE*, *OpenOffice.org*, *Apache*, *PostgreSQL*, *Firefox*, *Samba*, LaTeX etc.) ohne Aufwand installiert werden. Fertig kompilierte Pakete können von CD, DVD oder FTP-Servern installiert werden, alternativ können Pakete auch mit Hilfe von *pkgsrc* mit einem einfachen „make install“ selbst kompiliert werden.

32 Siehe auch http://www.pkgsrc.org.

5.3 pkgsrc als portable Lösung

Das Paketsystem von NetBSD – *pkgsrc* – stellt für die genannten und eine Reihe weiterer Probleme Lösungsmöglichkeiten bereit:

- *pkgsrc* ist ein System zum einfachen Installieren und Updaten von Paketen.
- *pkgsrc* ist ein Source-basiertes Paketverwaltungssystem.
- *pkgsrc* verwendet Original-Sourcecode zum Kompilieren.
- *pkgsrc* bietet die Möglichkeit, Binärpakete zu erzeugen und zu installieren.
- *pkgsrc* besteht aus zwei Komponenten, den Verwaltungstools und der eigentlichen Paketsammlung.
- *pkgsrc* behandelt Abhängigkeiten automatisch.

Ursprünglich von *FreeBSD* auf NetBSD portiert, hat *pkgsrc* heute als primäre Entwicklungsplattform NetBSD, wobei das System jedoch für viele weitere Systeme portiert wurde und auf diesen läuft: Unter anderem wurde *pkgsrc* für *AIX*, *BSD/OS*, *Darwin*, *FreeBSD*, *Irix*, *NetBSD*, *OpenBSD*, *Solaris* und *Dragonfly BSD* sowie auf die Linux Distributionen *SUSE*, *Debian*, *ROOT Linux*, *Slackware*, *RedHat*, *Fedora*, und *Mandrake* portiert.

5.4 pkgsrc in der Praxis

Die Installation von Paketen mittels *pkgsrc* ist denkbar einfach, um z. B. die grafische Benutzeroberfläche *KDE* mit all ihren Abhängigkeiten zu installieren, reichen folgende Befehle:

- cd pkgsrc/meta-pkgs/kde3
- make install

pkgsrc kann größtenteils ohne Root-Rechte betrieben werden, selbst umfangreiche Pakete wie *Firefox* oder *KDE* funktionieren ohne diese. Weiterhin kann das Verzeichnis, in das alle Pakete installiert werden sollen, frei gewählt werden. Die Standardeinstellung von „/usr/pkg“ ist für eine systemweite Installation sinnvoll, bei einer privaten Installation von Paketen ohne Root-Rechte ist oft das Heimatverzeichnis des Anwenders die bessere Wahl.

Neben dem Kompilieren von Quellcode – was sehr lange dauern kann – unterstützt *pkgsrc* auch das Erstellen von vorkompilierten Binärpaketen, die dann einfach mit *pkg_add(8)* installiert werden können.

Da Sicherheit auch bei *pkgsrc* groß geschrieben wird, ist mit dem Paket *audit-packages* die Möglichkeit vorhanden, über Sicherheitslücken und damit verbundene Updates informiert zu werden, um dann die entsprechenden Paketinstallationen zu aktualisieren.

Viermal jährlich wird ein Release von *pkgsrc*[33] erstellt, bei dem besonders darauf geachtet wird, dass alle Pakete optimal aufeinander abgestimmt sind; sie werden dann auch für die diversen Plattformen als fertige Binärpakete bereitgestellt.

6 Bleiben Sie am Ball!

pkgsrc wächst und gedeiht als Nebenprojekt von NetBSD seit 1997, NetBSD selbst seit März 1993 – länger als jede andere Alternative im Open-Source-Bereich.

Dank der Arbeit der Entwickler und dem Engagement vieler Anwender ist NetBSD heute stärker denn je, und die Entwicklungen gehen weiter: Die Unterstützung für Virtualisierungstechnologien wie *Xen* wird ständig verbessert, an verschiedenen neuen Dateisystemen wird intensiv gearbeitet, die NetBSD-Sicherheitsarchitektur wird weiter ausgebaut, und System- und Sicherheits-Updates werden zukünftig einfacher werden.

Das NetBSD-Projekt hat weiterhin eine interessante und starke Zukunft vor sich. Anwender haben die Gewissheit, dass ihr Betriebssystem von erfahrenen Programmierern weiterentwickelt wird und dass Innovationen beständig in das System einfließen. Weil das NetBSD-Projekt offen für Ansätze und Lösungen aus anderen Open-Source–Projekten ist, behalten die Anwender den Anschluss an die besten Entwicklungen der freien Software.

Hilfe bei Problemen ist schnell und unbürokratisch über Mailinglisten, IRC-Channel und über die Fehlerdatenbank möglich. Für formellere Unterstützung stehen auch die auf der NetBSD-Webseite aufgelisteten professionellen Berater zur Verfügung. Auch wenn keine Servicetelefonnummer existiert, werden Fragen vom NetBSD-Team selbstverständlich beantwortet – ohne dass man in einer Warteschleife hängt.

33 Zu finden unter http://ftp.netbsd.org/pub/pkgsrc/.

Flüssiger Code – Open-Source-Software in der Medienkunst

MARTIN PICHLMAIR

(CC-Lizenz siehe Seite 563)

Medienkunst steht in einem Nahverhältnis zur Technologie wie keine andere Kunstrichtung. Aus diesem Grund reagierte sie auch stark auf das Aufkommen von Open-Source-Software. Nicht nur gibt es einige Softwareprojekte mit Hilfe derer Kunst geschaffen wird, auch das Thema des freien Austausches von Wissen wurde vielfach reflektiert. Dieser Artikel stellt einige dieser künstlerischen Arbeiten vor und dokumentiert auch die Einbindung von Open-Source-Software in künstlerische Werke und deren Schaffungsprozess. Wesentliche Entwicklungen, die hierbei zu Tage treten sind die folgenden: Die Formbarkeit offener Software erlaubt eine Freiheit des künstlerischen Ausdrucks, der mit kommerziellen Programmen nur schwer zu erzielen ist. Die Produktionsbedingungen von Medienkünstlerinnen sind mit denen von Open-Source-Software einfach zu vereinen. So kann in beiden Fällen von einem ständigen *work in progress* gesprochen werden, wodurch die Tätigkeit des Schaffens ihr eigenes Gewicht gegenüber dem fertigen Produkt bzw. Werk besitzt. Der spielerische Zugang zur Technik erlaubt fundamentale Eingriffe in die Funktionsweise eines Systems. All dies trägt dazu bei, dass die Medienkünstler Code verwenden wie die Maler Farbe und die Bildhauer Ton. Sie gießen flüssigen Code nach ihren persönlichen Vorstellungen.

Schlüsselwörter: Kunst · Medienkunst · Pure Data · Tempest · Blender

1 Einleitung

Im Rahmen dieses Artikels wird ein Überblick über die Anwendung von Open–Source-Software (OSS) im Bereich der Kunst – hier vor allem den verschiedenen Spielarten der Medienkunst – gegeben. Aus verschiedenen Gründen ist OSS gerade im Bereich des künstlerischen und kulturellen Schaffens nicht mehr wegzudenken. Ei-

nerseits ist dieser Umstand sicher einfach in der Tatsache begründet, dass die Software frei erhältlich ist. Auf der anderen Seite scheint die Offenheit von Open-Source-Systemen verbunden mit ihrem handwerklichen Charakter, der die Arbeit am System stärker belohnt als die Anschaffung neuer Hardware, besonders gut zur Arbeitsweise von Künstlern zu passen. Des Weiteren zeichnet viele Open-Source-Projekte aus, dass sie durch ihre Offenheit nicht in so starkem Ausmaß ein bestimmtes Ergebnis nahelegen, wie dies bei proprietärer Software der Fall ist. Künstlerische Strategien bauen in manchen Fällen zum Beispiel darauf auf, gezielt Fehler zu verursachen und Software an ihre Grenzen zu treiben. Dieses Unterfangen ist im Falle proprietärer Software diametral zu den Interessen der Hersteller. Bei Open-Source-Software hingegen steht es dem Künstler frei, diese in beliebiger Weise zu formen und auch zu missbrauchen.

2 Überblick: Woher kommt die Medienkunst?

Es erscheint angebracht, vor der Betrachtung bestimmter Strategien in der Medienkunst zu klären, was mit letzterer denn gemeint ist. *Wikipedia* – eine verlässliche Quelle von Meinungen, eine unverlässliche Quelle von Wahrheiten – erklärt Medienkunst wie folgt:

> „New media art (also known as media art) is a generic term used to describe art related to, or created with, a technology invented or made widely available since the mid-20th Century. The term differentiates itself by its resulting cultural objects, which can be seen in opposition to those deriving from old media arts (i. e. traditional painting, sculpture, etc.) New Media concerns are often derived from the telecommunications, mass media and digital modes of delivery the artworks involve, with practices ranging from conceptual to virtual art, performance to installation." (Wikipedia 2006)

> „Medienkunst[1] ist ein generischer Begriff für Kunst, die mittels Technologien, die seit der Mitte des 20. Jahrhunderts erfunden wurden, geschaffen wurde oder auf diese referenziert. Der Begriff wird als Abgrenzung von der traditionellen Kunst (Malerei, Skulptur etc.) betrachtet. Des Öfteren sind die Themen der Medienkunst den Bereichen Telekommunikation, Massenmedien und digitale Distribution entnommen. Die künstlerische Praxis reicht von Konzeptkunst und Performance zu Installationen und Netzkunst."

Es gibt zahlreiche Blickwinkel, um die Ursprünge der Medienkunst zu betrachten. Vielfach werden Kunstsparten wie apparative Kunst oder Video- bzw. Filmkunst als

1 Die anglo-amerikanische Bezeichnung *new media art* bezeichnet zumeist digitale Medienkunst. Ergänzend dazu gibt es die Bezeichnung *media art* für Bereiche wie Videokunst, Tonbandmusik, analoge Videoinstallationen etc. Im deutschsprachigen Raum spricht man immer von *Medienkunst*.

zeitliche und inhaltliche Vorläufer erwähnt. Mit ersterer hat Medienkunst ihren Bezug zur Technik gemein, mit letzterer die Verwendung von (Massen-)Medien. Der Name „Medienkunst" rührt her von der Verwendung von Massenmedien in der Kunst. Die frühe Medienkunst entstand zu einer Zeit, in welcher der Medienbegriff – etwa durch die Werke von Marshall McLuhan – stark ausgeweitet wurde. Waren mit dem Begriff „die Medien" vor McLuhan nur Fernsehen, Journalismus, Radio und Kino gemeint, so wurden durch ihn Straßen, Geld und alle anderen Informationen transportierenden Mittel zu Medien. Ein früher Anspruch der Medienkunst war die Demokratisierung von Massenmedien nach der Hierarchisierung dieser durch die Propagandamaschinen des Zweiten Weltkriegs. Frühe Arbeiten des Medienkunstpioniers Peter Weibel nehmen aktiv auf diesen Wunsch Bezug. Guillaume Apollinaire hoffte schon in seinem 1913 erschienenen Buch über den Kubismus[2], dass mittels der Massenmedien die Isolation der Avantgarde überwunden werden kann, um „die Kunst und das Volk wieder miteinander zu versöhnen" (Apollinaire 1913, nach Daniels 2006).

Die Medienkunst stellt in diesem Sinne also eine Kunst dar, die auf Medien zurückgreift, um mit diesen über Medien Kunst zu machen. Und sie ist von Anfang an stark von dem Wunsch getrieben worden, die Medien mittels der dahinterstehenden Technologien dem Menschen zurückzugeben.

Der wohl wichtigste Text zur Medienkunst ist auch heute noch Walter Benjamins 1936 erschienener Essay[3] „Die Kunst im Zeitalter ihrer technischen Reproduzierbarkeit". In diesem programmatischen Text beschreibt Benjamin die Folgen der Ablösung des Originals (und den Verlust von dessen Aura) durch die technische Reproduzierbarkeit der Bilder. Obwohl inspiriert durch die Leistungen der Fotografie, erreichte dieser Text seine volle Tragweite erst durch die Entwicklung digitaler Techniken, denn mit diesen kam die Zeit der verlustfreien Kopie. Während das Original als Objekt (und Wert) natürlich unter einer perfekten Kopie leidet, stellt diese auf der anderen Seite die Basis der Open-Source-Software dar. Ebenso hat unbestritten die Distribution von Informationen, Musik und Video durch selbige einen gewaltigen Schub erhalten.

3 Open Source als Thema in der Medienkunst

Das Thema Open Source spielt in der Medienkunst in mehrerlei Hinsicht ein Rolle. Während die Verwendung von Open-Source-Software sicher den häufigsten Fall des Auftauchens dieses Begriffes darstellt, findet man bei genauem Hinsehen auch Werke, die den Gedanken der offenen Quelle thematisieren. Zwei prominente Beispiele sollen als Einführung in die Rolle von Open Source in der Medienkunst dienen: erstens das Werk *Temporary Autonomous Sarai (TAS)* des *Raqs Media Collective* und zweitens *Super Mario Clouds* von Cory Arcangel.

2 Siehe Apollinaire (1989), erstmals im Jahre 1913 veröffentlicht.
3 Erschienen in Benjamin (1987).

Temporary Autonomous Sarai ist eine temporäre Architektur, die einen sogenannten *sarai* in einer Ausstellung entstehen lassen soll. Die aus Indien stammenden *Raqs Media* beschreiben einen *sarai* mit den folgenden Worten:

> „The word *sarai*, or *caravansarai*, has its roots in Mughal India and refers to an enclosed space in a city or beside a highway—a destination and a point of departure—where travelers from all walks of life could find shelter, sustenance, and companionship." (nach Dietz 2003)

Der *sarai* ist also ein Treffpunkt des Austausches. Ein Knoten in einem Netz aus weiteren *sarai*. Der temporäre *sarai* wurde gemeinsam mit Studenten der *University of Minnesota* und dem japanischen Architekturbüro *Bow-Wow* erstellt. Er vereint den Gedanken des Treffpunkts von Menschen mit dem des Treffpunkts von Ideen. Besucher der Ausstellung, in der die *TAS* gezeigt wurde, konnten an diesem Ort Nachrichten hinterlassen, Nachrichten ihrer Vorgänger editieren und so den Raum zur Kollaboration nutzen. Sie konnten die vorhandene Architektur als Informationsarchitektur verwenden.

Noch näher an die Welt der Open-Source-Software führt *Super Mario Clouds* heran, eine Installation, die von Cory Arcangel etwa 2002 ins Leben gerufen wurde. Die Arbeit ist eine Veränderung des Spiels *Super Mario* von *Nintendo*. Die Veränderung besteht in diesem Fall daraus, dass alle Inhalte des Spiels mit Ausnahme der Wolken und des Himmels entfernt wurden. Selbst in dieser kondensierten Form erkennt jeder Besitzer des Spiels es sofort wieder. Dieser Eingriff in den geschlossenen Code war durch illegales Disassemblieren des Spiels möglich. Dieses Vorgehen wird als künstlerische Strategie Teil des Werks. So stellte Cory Arcangel noch bevor er sein Werk ausstellte ein Howto auf seine Website, in dem er geradezu akribisch erklärte, wie man das Spiel *Super Mario* selbst verändern kann: „Below I will go through the source code line by line" (Arcangel 2005). Sein Howto beschreibt den Assemblercode, den er veränderte, enthält den neuen Code und eine in 17 Schritten verfasste Anleitung (mit Bildern), wie man den mit dem veränderten Code bespielten Chip in das *Cartridge* vorher aus- und nachher wieder einbaut. Zuzüglich zur Veröffentlichung im Web findet sich diese Anleitung auch in einem bei Ausstellungen ausliegenden Folder. Cory Arcangel thematisiert so die Widersprüche zwischen geschlossenem und offenem Code. Wer die Arbeit Arcangels sieht, kann also nach Hause gehen und dort selbst die Arbeit nachbauen oder eine neue Version der Arbeit erstellen:

> „Hey, welcome to the 'Don't Touch My Computer Home Users Guide' a pdf 4 my first art show in Paris!!!! [. . .] basically this is a guide to help you better to understand the technique of how I made the works in the show, and that will allow you to recreate them at home." (Arcangel 2005)

> „Hallo, willkommen zum 'Don't Touch My Computer Home Users Guide', einem PDF zu meiner ersten Ausstellung in Paris!!!! [. . .] im

> Grunde genommen ist dieser Guide dazu da, dir zu helfen, die Techniken zu verstehen, mit denen ich die Arbeiten dieser Ausstellung gemacht habe. Und das wird dir erlauben, sie zu Hause nachzubauen."

Beide hier besprochenen künstlerischen Arbeiten thematisieren zentrale Aspekte der Open-Source-Bewegung. Zum einen behandeln sie kulturelle Produkte (Code, Texte, Bilder, Architektur) als veränderbar, ja sie fordern sogar zur eigenmächtigen Veränderung dieser auf. Zum anderen reflektieren beide in unterschiedlicher Weise über die sozialen Zusammenhänge dieser Verformbarkeit. *TAS* baut einen Raum, welcher der Kommunikation dient und es erlaubt im Zuge dessen, auch rekonfigurativ tätig zu sein. *Super Mario Clouds* macht den Ausstellungszuschauer zum potenziellen Hacker.

Dieses Aufbrechen bestehender Technologien – es muss ja nicht immer ein illegaler Hack sein – ist teilweise auch einfach vom Wunsch getrieben, bestimmte Funktionalitäten zu realisieren. Geschlossene Systeme bieten eben nicht den beliebigen Zugriff auf ihre Subfunktionen. Da aber die Kunst im Allgemeinen von Visionen und Vorstellungen über eine bestimmte zu realisierende Situation (das mag ein Bild, eine Installation oder auch ein Gerät sein) bestimmt ist, sind manche Künstler nahezu gezwungen, auf offene Systeme zurückzugreifen. Dieses Problem trat auch im Zuge der Arbeit an den *Seven Mile Boots* (Pichlmair 2005, siehe auch Abbildung 1) – einem „Wearable Art Piece" bestehend aus zwei Paar elektronischen Stiefeln – auf. Die *Siebenmeilenstiefel*, die gemeinsam mit Laura Beloff und Erich Berger realisiert wurden, sind intelligente Schuhe. Sie ermöglichen dem, der sie angezogen hat, durch das Internet zu gehen. Je nachdem, wie sich die Trägerin bewegt, werden einzelne Chaträume betreten. Das dort Gesprochene wird in Sprache übertragen, die dann durch in die Stiefelspitzen eingebaute Lautsprecher ausgegeben wird. Wenngleich es zu Beginn der Planung des Projekts im Jahr 2002 schon industrielle Lösungen gab, die von uns hätten genutzt werden können, war es eindeutig einfacher und billiger für uns, einen PDA, der für *Microsoft Windows CE* konzipiert war, unter *Linux* zu verwenden. Der Einsatz des offenen Betriebssystems erlaubte es nicht nur, eine komfortable Entwicklungsumgebung zusammenzustellen, sondern auch, auf sehr hardwarenahe Funktionen wie den Ladestand der Batterie auf sehr einfache Art und Weise zuzugreifen. Es waren zu vorderst technische Überlegungen, die zu der Entscheidung für *Linux* führten. Aber natürlich spielte es auch eine Rolle, dass das Team über mehr Erfahrung mit dieser Plattform als mit dem proprietären Äquivalent verfügte. Für den Ausstellungsbesucher war die technische Basis nicht einsichtig.

4 Die technologische Seite

Es ist eine der Grundfesten der Medienkunst, dass sie nicht nur versucht, ein Kunstwerk mit Hilfe neuer Medien zu erstellen (sonst wären viele Hollywood-Filme Medienkunst), sondern dass sie diese auch kommentiert und diskutiert. Medien stellen also

Abbildung 1: Die „Seven Mile Boots" von Laura Beloff, Erich Berger und Martin Pichlmair verwenden nur Open-Source-Software (z. B. Perl, flite, Linux, Audacity, sox, GCC).

in gleicher Weise das reflektierte Thema wie die Technik dar. Diese etwas introspektive Herangehensweise voller Referenzen ist nicht zuletzt deshalb in der Medienkunst weniger problematisch als in anderen Kunstrichtungen, weil unser Alltag so stark von Medien geprägt ist. Während ein Bild über Bilder sehr weit von alltäglichen Eindrücken der meisten Menschen entfernt ist, kann ein Fernsehfilm, der auf Fernsehen verweist, persönlich leicht erfasst werden.

Zumeist wird Open-Source-Software aus technischen und nicht aus inhaltlichen Überlegungen verwendet. Anders als in den beiden einleitend beschriebenen Arbeiten ist sie dann sehr oft versteckt. Ein Werk könnte auf Basis von *Linux* arbeiten oder einfach auf einem Windows-Computer. Ein Musikstück könnte entweder mittels freier Software erstellt worden sein oder mit kommerzieller. Im Folgenden sollen einige Werke vorgestellt werden, die einerseits auf Open-Source-Software als Technologie aufbauen und andererseits diesen Umstand auch auf die eine oder andere Weise sichtbar machen.

Ein poetisches Beispiel für das Nach-außen-kehren von inhärenten Eigenschaften eines Mediums stellt *JULU*[4] von Alan Sondheim dar. *JULU* ist ein in *Perl* verfasstes interaktives Gedicht. Das Gedicht-Skript wird gestartet und nach einigen Zeilen Text fordert es den Leser auf, die Frage „Your stitch should be wiped into existence?" zu beantworten. Die Antwort darauf und auf weitere kommende Fragen fließt in die weitere Ausführung des Codes ein. Darüber hinaus ist *JULU* nicht fehlerfrei programmiert.

4 Siehe http://runme.org/project/+sondheimjulu.

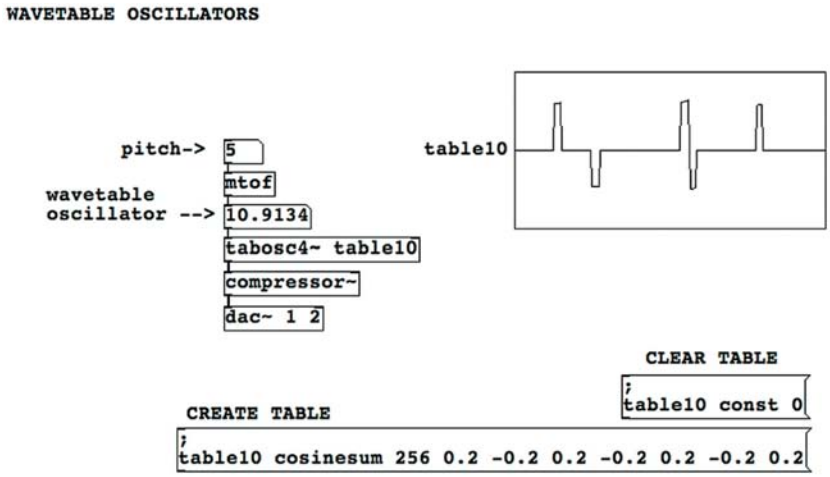


Abbildung 2: Ein „Patch" genanntes Stück Code in Pure Data

Man kann der Tatsache, dass sich zum Beispiel die Fehlermeldung „Use of uninitialized value in concatenation (.) or string at sondheim-julu.pl line 129, <STDIN> line 8." in den Output des Gedichts drängt, durchaus Absicht unterstellen. Aber nicht nur das ausgeführte Skript, auch der Code selbst liest sich wie ein Gedicht. Die Vorstellung, dass Code ästhetischen Wert besitzt, ist eine, die so mancher Vertreter der Open–Source-Bewegung teilt. Denn veröffentlichter Code wird von der Community nicht nur danach beurteilt, ob und wie er funktioniert, sondern auch danach, wie er aussieht, aufgebaut und erweiterbar ist, kurz: wie er sich anfühlt. Die Ästhetik von Code als Form der interaktiven Dichtung ist also eine, die dem Open-Source-Gedanken nahe steht.

4.1 Pure Data

Eines der am häufigsten verwendeten Open-Source-Programme im Bereich der Kunst ist *Pure Data*[5]. Die Geschichte dieser Software geht auf das Jahr 1986 zurück. Damals entwarf Miller Puckette *Patcher*, eine Art Simulation eines Synthesizers in Software (Puckette 1988). Während das Programm einige technologische Umstellungen erfuhr, dank derer sich auch sein Name immer wieder änderte, blieb der Kern der Software immer der gleiche: eine Oberfläche, die Musikern das Programmieren ermöglicht (siehe Abbildung 2). Im Jahr 1996 teilte sich das inzwischen *Max* genannte Projekt. Während die Firma *Cycling '74* bis heute *Max/MSP* kommerziell vertreibt, startete Miller Puckette ein unter der BSD-Lizenz stehendes eigenes Projekt, genannt *Pure*

5 Download unter: http://lena.ucsd.edu/~msp/software.html.

Data oder einfach *Pd* (Puckette 1996). Neben *Pure Data* und *Max/MSP* wurde am französischen *Institut de Recherche et Coordination Acoustique/Musique* mit *jMax* eine weitere Open-Source-Software zum visuellen Programmieren von Musik entwickelt. Letztere wurde etwa von so bekannten Komponisten wie Pierre Boulez oder Philippe Manoury verwendet. In einem persönlichen Gespräch verglich Miller Puckette *Pure Data* einmal mit einer Leinwand, auf die der Musiker Code aufträgt wie ein Maler Farbe.

Einer der Hauptgründe, warum *Pd* sich einer stetig wachsenden Benutzerschaft erfreut, ist die unendliche Anzahl an Möglichkeiten, die es dem Musiker eröffnet. Durch seine grafische Programmieroberfläche erschließt sich die Software – für Musiker sogar eher als für Programmierer – sofort. Dennoch erhält der Benutzer die vollen Möglichkeiten einer „echten" Programmiersprache. Den Kern der Anwendung stellt ein modularer Synthesizer dar. Dieser wurde nach und nach unter anderem mit Funktionen zur Echtzeitbearbeitung von Video, 3D-Grafik, *physical modelling* und Sensordaten bereichert. Kontemporäre Beispiele für die Verwendung von *Pure Data* sind etwa: *TEMPEST* von Erich Berger (Abbildung 3), *reacTable* (ein haptisches Musikinstrument, siehe Jorda et al. 1987) oder die Installation *resonanCITY* von Sara Kolster und Derek Holzer. Letzterer sagte einmal in einem Interview:

> „I used a lot of cracked commercial software for a lot of years when doing sound and I always got a couple of feelings out of it. One feeling was that you get these fancy programs with these fancy user interfaces, but at the end the more they have created this environment that's very easy for you to use, the more they've actually determined the kind of work you can make with it. If you look at a program like *Ableton Live* which is used by probably about eighty percent of people making sound and performing out live these days, it seems like it's good for a very few things, it's good for working with loops, putting effects on these loops and sequencing them, but it pushes you in one creative direction, it pushes you into making a certain kind of music, really it pushes you towards German techno more than anything else." (Derek Holzer nach Mattin 2004)

> „Ich verwendete viele Jahre lang mit gemischten Gefühlen geknackte kommerzielle Software. Einer der Eindrücke, die ich davon hatte war der, dass man diese schicken Programme mit ihren schicken User Interfaces bekommt, die je mehr sie diese Umgebung bieten, desto mehr deine Möglichkeiten vorherbestimmen. Wenn du Programme wie *Ableton Live*, welches von vielleicht achtzig Prozent der Leute, die heute Musik machen und live spielen verwendet wird, betrachtest, scheint es nur für sehr wenige Dinge gut zu sein. Es ist gut für die Arbeit mit Loops, dafür, Effekte auf diese Loops zu legen und sie zu sequenzieren, aber

> es leitet dich kreativ in eine bestimmte Richtung. Es verleitet dich dazu, eine bestimmte Art Musik zu machen. Mehr als alles andere verleitet es dich dazu, deutschen Techno zu machen."

Der Musiker Patrick King geht so weit, zu behaupten:

> „'Made by hackers, for hackers', *Linux* philosophically is at odds with proprietary operating systems, it's user contributed, it's free (as in speech and beer), comes in many forms and flavours and is fast, stable and usually won't break without human error." (Patrick King nach Poole 2004)

> „'Gemacht von Hackern, für Hacker' befindet sich *Linux* philosophisch im Konflikt mit proprietärer Software. Es wird von den Beiträgen der Benutzer getragen, es ist frei (wie in Rede und Bier), ist in vielfältigen Formen und Geschmacksrichtungen erhältlich, ist schnell, stabil und geht zumeist nur durch menschliche Fehler kaputt."

Open-Source-Software erlaubt also dem Künstler ein größeres Spektrum an Ausdrucksmöglichkeiten. Sie legt nicht so sehr eine bestimmte ästhetische Form nahe wie proprietäre Software. Und selbst wenn sie dies täte, ließe sie sich sofort verändern. Sie würde sich formen und biegen lassen. Gerade diese Freiheit ist es aber, die auch einschränkend wirkt: Einerseits verlangt sie häufig Expertenwissen und andererseits sind gewisse Schranken der Möglichkeiten oftmals durchaus förderlich für die Kreativität. Die Angst des Autors vor dem leeren Blatt Papier ist eine Angst vor der Unendlichkeit der Möglichkeiten. Solche Herausforderungen können nur durch Lernen überwunden werden.

Linux (und in dessen Schlepptau hunderte Applikationen) läuft nicht nur auf Computern, sondern darüber hinaus auf Telefonen, PDAs, (mobilen) Spielekonsolen und Netzwerkroutern. All diese Geräte werden durch das freie Betriebssystem leichter und uniformer zugänglich. Code kann einfacher portiert werden, wenn ein *GNU Compiler* (oder auch *Perl* oder *Python*) für den Host verfügbar ist. Natürlich ist dies auch nur möglich, wenn die Quellen vorliegen. Ebenso ist der Zugriff auf Hardware im Grunde genommen auf allen Linux-Systemen identisch, wodurch die Entwicklung zusätzlich erleichtert wird. Dies erlaubt, mit ihnen zu spielen und diese Geräte einer künstlerischen Nutzung zu öffnen. Genau dafür erhielt Linus Torvalds auch stellvertretend für die ganze Entwicklergemeinschaft auf der *Ars Electronica 1999* in der Kategorie *.net* eine goldene *Nica*. Diesem Umstand ist *Processing*[6] zu verdanken, ein weiteres Beispiel von Open-Source-Software in der Kunst.

6 Download unter: http://processing.org.

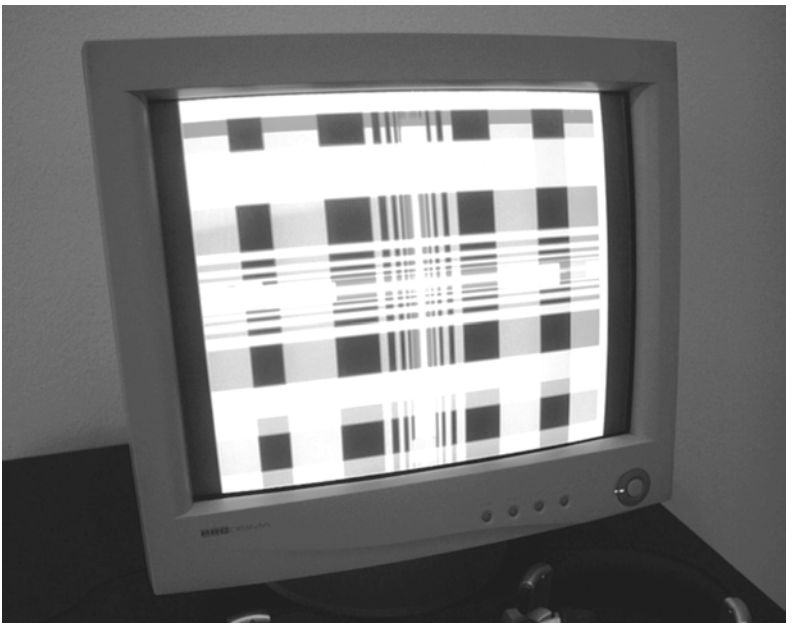


Abbildung 3: TEMPEST von Erich Berger ist vollständig mittels Pure Data realisiert. (Foto: José Luis de Vicente, CC-BY-NC-SA)

4.2 Processing und Blender

Processing, ein von Ben Fry und Casey Reas gestartetes Projekt, ist eine Java-Entwicklungsumgebung, die sich besonders an Nicht-Programmierer wendet. Grafiker und Videokünstler können mittels *Processing* interaktive (oder auch nur datenintensive) Visualisierungen und künstlerische Arbeiten erstellen (siehe Abbildung 4). Die Software wurde unter anderem für die Lehre entwickelt. *Processing* erleichtert den Einstieg in das Programmieren, indem es sowohl prozedurales als auch objektorientiertes Coden erlaubt. So können künstlerische Projekte mit dem Künstler wachsen, ohne dass dieser neue Software lernen muss. Hinzu kommt die Tatsache, dass *Processing* – ebenso wie *Pure Data* – natürlich auf allen großen Plattformen läuft. Seit kurzem existiert sogar eine Processing-Version für das Series60-Betriebssystem von *Symbian*. Damit ist eine Software entstanden, die es ermöglicht, das kreative Potenzial dieser jungen Plattform freizulegen, ohne sich dabei den Zwängen proprietärer Software unterzuordnen.

Im 3D-Bereich stellt *Blender* wohl eines der herausragendsten Beispiele für erfolgreiche Open-Source-Projekte dar. Während *Blender* ursprünglich eine *3D authoring software* war, ist auch dieses Programm heute gewachsen. So wurde es im Laufe der

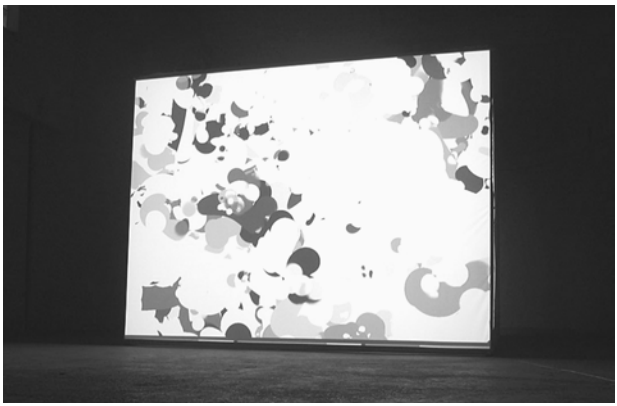


Abbildung 4: Process 8 (Software 2) von C. E. B Reas, erstellt mittels Processing. (Photo: Marius Watz, CC-BY-NC-SA)

Zeit erweitert, bis es eine komplette *game engine* aufweisen konnte. *Blender* entstand als In-House-Projekt der holländischen Animationsfirmen *NeoGeo* und *Not A Number Technologies*. Als *Not A Number* 2002 Konkurs anmeldete, wurde der Source-Code von *Blender* von dessen ursprünglichem Autor, Ton Roosendaal, mit Hilfe von Spenden aus der Konkursmasse gekauft und unter der *GNU General Public License* veröffentlicht. *Blender* ist eine 3D-Modelling-Umgebung, die sich mittlerweile mit proprietären Systemen wie *Maya* und *3D Studio* messen kann. Wie auch *Processing* und *Pure Data* wird *Blender* erfolgreich im kommerziellen Bereich eingesetzt. Ein bekannter Fall ist die Verwendung von *Blender* zur Prävisualisierung in dem Hollywoodfilm *Spiderman 2* des Regisseurs Sam Raimi.

5 Resümee

Die Anzahl der Künstler, die Open-Source-Software in ihren Medienkunstprojekten verwenden, ist gewaltig. Auch das Gebiet, das dabei technologisch, aber auch inhaltlich abgedeckt wird, ist schier unüberschaubar. Im Rahmen des vorliegenden Textes konnten nur einige kurze Schlaglichter auf dieses Gebiet gerichtet werden. Tools wie *Processing*, *Blender*, *Linux* und *Pure Data* erleichtern die Verfolgung künstlerischer Strategien, indem sie technische Geräte öffnen und zugänglich machen. So profitiert die Kunst von der Open-Source-Bewegung.

Auf der anderen Seite geben die Künstler auch etwas an diese Software(-kultur) zurück: Durch ihre Arbeiten machen sie Technologie sichtbar, betastbar und ästhetisch wahrnehmbar. Medienkünstler formen Medientechnologien wie Bildhauer Ton: Sie kneten und gießen Code. Dieser spielerische Umgang mit Hardware und Software

passt nur zu gut zur Welt der Open-Source-Entwickler: einer Welt, in der Code schon immer anpassbar war, den ganz persönlichen Bedürfnissen unterworfen und der Öffentlichkeit präsentiert wurde. Einer Welt des flüssigen Codes.

Literatur

Apollinaire, G. (1989), *Die Maler des Kubismus*, Luchterhand Literaturverlag, Frankfurt.

Arcangel, C. (2005), 'Super Mario Clouds – 2005 rewrite', Werkbeschreibung des Künstlers. http://beigerecords.com/cory/Things_I_Made_in_2003/mario_clouds_2005.html [13. Dez 2006].

Benjamin, W. (1987), *Abhandlungen*, Bd. I (2. Teilbd.) aus *Gesammelte Schriften*, Suhrkamp Verlag, Frankfurt am Main, Kapitel „Das Kunstwerk im Zeitalter seiner technischen Reproduzierbarkeit", S. 501.

Daniels, D. (2006), 'Medienkunst im Überblick: Vorläufer'. http://medienkunstnetz.de/themen/medienkunst_im_ueberblick/vorlaeufer/ [13. Dez 2006].

Dietz, S. (2003), 'Raqs Media Collective', Ausstellungsbeschreibung. http://latitudes.walkerart.org/artists/index.wac?id=80 [13. Dez 2006].

Jorda, S., Kaltenbrunner, M., Geiger, G. und Bencina, R. (1987), The reacTable, *in* 'Proceedings of the ICMC', International Computer Music Association.

Mattin, M. (2004), 'Give It All, Zero For Rules!'. http://www.metamute.com/look/article.tpl?IdLanguage=1&IdPublication=1&NrIssue=24&NrSection=5&NrArticle=1502 [13. Dez 2006].

Pichlmair, M. (2005), Seven Mile Boots: Implications of an Everyday Interface, *in* 'Proceedings of the 13th annual ACM International Conference on Multimedia', S. 574–577.

Poole, J. (2004), 'Pure Data in the Tropics: Patrick King'. http://www.octapod.org:8000/jeanpoole//archives/2004_05.html [05. Jan 2007].

Puckette, M. (1988), The Patcher, *in* 'Proceedings, International Computer Music Conference', International Computer Music Association, San Francisco, S. 420–429.

Puckette, M. (1996), Pure Data, *in* 'Proceedings, International Computer Music Conference', International Computer Music Association, San Francisco, S. 420–429.

Wikipedia (2006), 'New Media Art'. http://en.wikipedia.org/wiki/New_media_art [13. Dez 2006].

Kapitel 5

Open Source bildet

„Ohne Bildung sehen wir den Fortschritt der Menschheit als Zuschauer, auf den Bildschirmen, die es immer geben wird, aber wir werden weder Handelnde noch Teilnehmer sein."

– *Sergio Ramirez*

Open Source bildet

Silvia Varadinova und Hans Henning

(CC-Lizenz siehe Seite 563)

In einer bekannten Skizze benennt Lawrence Lessig vier unterschiedliche Faktoren, die das Verhalten von Menschen in unserer Gesellschaft beeinflussen: Normen, Gesetze, Markt und Architektur. In unserer modernen IT-dominierten Welt wird diese Architektur in vielen Fällen durch Computersysteme bestimmt. Die Fähigkeit zum Umgang mit dieser Architektur ist in unserer Gesellschaft nicht nur sinnvoll, sondern geradezu überlebensnotwendig, um nicht wie im Eingangszitat angedeutet zu einem bloßen Zuschauer künftiger Entwicklungen zu werden. Der Vermittlung dieser Fähigkeit sollte demnach bereits in der Ausbildung ein entsprechend hoher Stellenwert eingeräumt werden. Das vorliegende Kapitel widmet sich dem Thema „Open Source in der Bildung" und versucht aufzuzeigen, welche Potenziale der Einsatz von Open Source bei der Vermittlung von IT-Kompetenz haben kann.

Den Einstieg bietet Christina Dörges Artikel über den „Einsatz von Open-Source–Software (OSS) zur Vermittlung von IT-Schlüsselqualifikationen". Darin stellt sie ein Unterrichtskonzept vor, welches auf dem konzeptionellen Verständnis technischer Sachverhalte aufbaut. In unserer Zeit, wo die Entwicklung im technischen Feld rasch ist und der Computer eine so wichtige Rolle im Privat- und Berufsleben einnimmt, ist das Verständnis von Basiskonzepten, auf denen Computer und ihre Anwendungen beruhen, zukunftswichtig. Nur wer diese Schlüsselqualifikationen beherrscht, kann das erworbene Wissen auf neue Situationen anwenden und sich so selbstständig neue IT-Systeme erschließen, so dass er von einem einfachen Konsument zu einem handlungsfähigen Nutzer wird.

Während sich Frau Dörge in ihrem Artikel weitgehend auf die Erfahrungen mit ihren Lehramtsstudenten bezieht, setzt Dr. Herbert Reckmann mit seinem Artikel „Pädagogische und gesellschaftliche Potenziale freier Software – GNU/Linux in Schule und Elternhaus" den Fokus auf den Einsatz von Open-Source-Software an Schulen. Am Beispiel der inzwischen in vielen Klassenräumen vorhandenen Medienecke wird auf die Möglichkeiten eingegangen, die der Einsatz von IT im Unterrichtsgeschehen bietet. Dabei werden gleichzeitig die möglichen Vorteile von OSS gegenüber proprietärer Software im Schuleinsatz dargestellt. Zusätzlich setzt der Autor einen starken

Akzent auf die gesellschaftlich wertvollen und medienerzieherischen Dimensionen, die OSS den Schülern bietet. Durch die offene Kommunikation und den freien Austausch von Wissen werden die Schüler für Themen wie freie Meinungsäußerung und informationelle Selbstbestimmung sensibilisiert.

Auch Christian Dorninger geht mit seinem Artikel „Zwei IT-Welten für alle SchülerInnen – Schule und freie Software" auf das Thema „OSS an Schulen" ein, und beschreibt die Open-Source-Szene an den österreichischen Schulen. Neben dem Einsatz von freier und Open-Source-Software in der Bildung selbst geht er dabei auch auf Open-Source-Lösungen zur Unterrichtsverwaltung ein und zeigt darüber hinaus, wie das Open-Source-Prinzip auf den Lernprozess angewendet wird.

Von der technisch gut ausgestatteten Bildungswelt Westeuropas macht der Artikel von Nazir Peroz, Jan Suhr und Daniel Tippmann einen Sprung nach Afghanistan. In „Einsatz von Open-Source-Software in der Entwicklungszusammenarbeit – das Beispiel Afghanistan" beschreiben die Autoren ihre Erfahrungen beim Aufbau von IT-Infrastrukturen an den Universitäten von Kabul und Herat und gehen dabei auch auf die Potenziale ein, die der Einsatz von Open-Source-Software für Entwicklungsländer bietet, insbesondere auf die Chancen beim Aufbau akademischer Strukturen.

Gerade am Beispiel von Afghanistan, einem Land, das kaum Geld für die Bildung übrig hat, zeigt sich, wie sinnvoll der Einsatz von Open Source sein kann. Die finanziellen Aspekte stellen jedoch nur einen Teil der Möglichkeiten von Open Source dar.

In allen Artikeln dieses Kapitels wird dargelegt, wie das Open-Source-Prinzip neue Ideen und Strukturen für die Lehre entwickelt, denn die freie Weitergabe von Wissen ist grundlegend für fast alle Lernprozesse. Bildung beeinflusst die Art und Weise, in der Menschen ihre Umwelt wahrnehmen und mit dieser interagieren. Dadurch stellt Bildung eine Art Meta-Faktor dar, der neben den eingangs erwähnten vier Faktoren eine Sonderstellung einnimmt: Erst durch Bildung wird es den Menschen möglich, Normen, Markt, Architektur und Gesetze zu verstehen, zu bewerten und ihr Handeln danach auszurichten oder diese kritisch zu hinterfragen.

Einsatz von Open-Source-Software zur Vermittlung von IT-Schlüsselqualifikationen

CHRISTINA DÖRGE

(CC-Lizenz siehe Seite 563)

IT-Schlüsselqualifikationen unterrichten bedeutet, die Konzepte zu erklären, auf denen Computer und ihre Anwendungen beruhen, anstatt reine Anwenderschulungen für bestimmte Programme durchzuführen. Die Vermittlung von IT-Schlüsselqualifikationen soll die Lernenden befähigen, erworbenes Wissen auch auf neue Anwendungen und neue Situationen anwenden zu können. Sie sollen somit handlungsfähige Nutzer werden, statt reine Konsumenten. Dieses Unterrichtskonzept wurde von uns an der Universität Bremen in der Arbeitsgruppe „Digitale Medien in der Bildung“ (DiMeB) mit Hilfe von Open-Source-Software entwickelt und umgesetzt. Erste Erfahrungen zeigen interessante Ergebnisse und eine vielversprechende Richtung für weitere Schulungen auf.

Schlüsselwörter: Schlüsselqualifikation · Lehrerausbildung · Alltagsalgorithmen

1 Einleitung

Der Computer nimmt in unserer Gesellschaft eine immer dominantere Rolle ein, sei es in unserem Privatleben oder im Berufsalltag. Jedoch vollzieht sich die Entwicklung gerade im technischen Feld so schnell, dass meist nur wenige Personen in der Lage sind, sich über IT-Themen eine qualifizierte Meinung zu bilden.

IT-Schlüsselqualifikationen sollen solcher Unkenntnis entgegenwirken, indem sie eine Grundlage schaffen, auf der man sich selber Wissen aneignen kann. Sie stehen damit im Gegensatz zu den häufig angebotenen Anwenderschulungen, welche kein konzeptionelles Verständnis technischer Sachverhalte lehren.

In diesem Aufsatz möchte ich anhand von Erfahrungen in der universitären Lehre zeigen, welche Rolle Open-Source-Software bei der Vermittlung von IT-Schlüsselqualifikationen spielen und welche Auswirkungen der Einsatz in Schulungen und Ausbildung haben kann.

Für einen großen Teil der Softwarenutzer steht zuerst der Aspekt der Kostenlosigkeit von Freier/Open-Source-Software (OSS) im Vordergrund. OSS-Produkte bieten aber darüber hinaus weitere Vorteile, die besonders bei Schulungen von IT-Schlüsselqualifikationen von großem Nutzen sein können:

- Eine Vielzahl von sich in Funktion und Design unterscheidenden Anwendungen
- Leichte Zugänglichkeit (etwa durch das Internet oder Zeitschriften-CDs)
- Anpassbarkeit durch Anwender mit entsprechenden Fähigkeiten
- Die Existenz von Communitys, in denen diese Software entwickelt wird und die als Ansprechpartner zur Verfügung stehen

2 Schlüsselqualifikationen im Bereich IT

Der Begriff der Schlüsselqualifikationen wurde 1974 von Dieter Mertens, dem damaligen Leiter des Nürnberger Instituts für Arbeitsmarkt- und Berufsforschung, geprägt. Er unterschied vier Arten von Schlüsselqualifikationen (vgl. Mertens 1974, S. 36):

1. Basisqualifikationen (umfassen etwa strukturiertes und logisches Denken)
2. Horizontalqualifikationen (etwa die Transferfähigkeit von einer bereits erlernten Fremdsprache in eine andere)
3. Breitenelemente (spezielle Kenntnisse und Fertigkeiten, die berufsübergreifend benötigt werden, wie etwa die Grundrechenarten)
4. Vintage-Faktoren (Bildungsdifferenzen aufgrund des Alters)

In ihrem Leitfaden „Schlüsselqualifikationen – Wie vermittele ich sie Lehrlingen?“ führen die beiden Autorinnen Sabine Archan und Elisabeth Tuschek ihrerseits drei Arten von Schlüsselqualifikationen auf (Archan und Tuschek 2002, S. 5):

1. Sachkompetenz (Schlüsselqualifikationen im Umgang mit Sachen und Problemen)
2. Sozialkompetenz (Schlüsselqualifikationen im Umgang mit anderen)
3. Selbstkompetenz (Schlüsselqualifikationen im Umgang mit sich selbst)

Der Kernaspekt der Schlüsselqualifikationen, der von allen Autoren beschrieben wird, ist die Erschließung von Spezialwissen, das über Faktenwissen hinausgeht und Menschen befähigen soll, sich auch auf eine „nicht prognostizierbare Zukunft“ vorbereiten zu können.[1]

1 Die folgenden Autoren wurden herangezogen: Archan und Tuschek (2002, S. 3), Gonon (2002, S. 6), Mertens (1974, S. 36, 41 f.).

Didi et al. haben alle Schlüsselqualifikationsbezeichnungen zusammengetragen und sind dabei allein in der berufspädagogischen Literatur auf über 600 gestoßen (Didi et al. 1993). Sie haben die zehn meistgenannten Schlüsselqualifikationen aufgelistet, die Helen Orth in ihrem Buch wiedergibt (Orth 1999, S. 2):

- Kommunikationsfähigkeit
- Kooperationsfähigkeit
- Flexibilität
- Kreativität
- Denken in Zusammenhängen
- Selbständigkeit
- Problemlösefähigkeit
- Transferfähigkeit
- Lernbereitschaft
- Durchsetzungsfähigkeit

Dass die Unterteilung und Bezeichnung von Schlüsselqualifikationen nicht einheitlich ist, mag darin begründet liegen, dass die Forschungsgebiete der Autoren ein sehr breit gestreutes Feld umfassen. Daher wird der Kernaspekt der Schlüsselqualifikationen von jedem Autor entsprechend seines Fachgebiets unterschiedlich interpretiert und konkretisiert werden. Als Grundlage wird jedoch immer angenommen, dass es um die Vermittlung funktions- und berufsübergreifender Fähigkeiten geht, welche als Schlüssel zu weiteren Qualifikationen dienen können.

3 IT-Schlüsselqualifikationen in der Berufspädagogik

Es ist unsere Überzeugung, dass Menschen auf dem Gebiet der Informationstechnik handlungsfähig gemacht werden müssen, damit sie in der Lage sind, sich den heutigen Entwicklungen anpassen zu können. Da zielgerechtes Handeln in komplexen Umgebungen konzeptionelles Verstehen voraussetzt, spielen IT-Schlüsselqualifikationen hier eine entscheidende Rolle.

Bei der Diskussion um die Allgemeinbildung forderte der Erziehungswissenschaftler Klafki schon 1985 eine größere technische Allgemeinbildung, da er davon überzeugt war, dass Änderungen und Neuerungen im technischen Bereich stets bedeutende Veränderungen von Anforderungen an die Bevölkerung nach sich ziehen (Klafki 1985).

Grundsätzlich sollten zwei Arten der Vermittlung von IT-Wissen unterschieden werden (Dörge und Büschenfeldt 2006, S. 4):

1. IT-Wissen in Form von Anwenderwissen bzw. Bedienerschulung für eine bestimmte Software

2. IT-Wissen, welches den Menschen hilft, sich ihre Technikumwelt selber zu erschließen und sich den stetig wandelnden Bedürfnissen anzupassen

Zwar hat sich inzwischen in der Bevölkerung und in den Schulen durchgesetzt, dass Computerkenntnisse wichtig und sinnvoll sind. Aber leider wird in erster Linie Anwenderwissen vermittelt, anstatt Konzeptschulungen durchzuführen. Die amerikanische *Computer Science Teachers Association (CSTA)* fordert in ihrem Bericht von 2005, Computernutzungsfähigkeiten nicht darauf zu beschränken, den Rechner als Hilfsmittel für andere Disziplinen einzusetzen, sondern als eigenständige Disziplin zu sehen (vgl. Stephenson et al. 2005, S. 14)

Folgende Aspekte sollten daher unbedingt in Kurskonzepte zur Vermittlung von IT-Schlüsselqualifikationen eingebracht werden (Dörge und Büschenfeldt 2006, S. 5):

- Selbständiges Finden von eigenen Wegen zur Lösung von Problemen durch die Nutzung digitaler Medien
- Selbständiges Einarbeiten in Folgeversionen von Software oder in alternative Programme
- Eigenständig den Wissensstand aktuell halten
- Nutzungsmöglichkeiten von IT für die eigenen Bedürfnisse sehen
- Gefahren und Risiken für das eigene System erkennen können
- Trends und Neuentwicklungen nutzen
- Die gesellschaftliche Relevanz und Auswirkungen einschätzen können (Potenziale wie Gefährdungen)

Zusätzlich ist zu beachten, dass Studierende in der Regel bereits mit einigen Schlüsselqualifikationen ausgestattet sind. Zu diesen gehören die Fähigkeit zur Kooperation, Kommunikationsfähigkeiten sowie logisches und strukturiertes Denken. Aus diesem Grund sollten „aktivierende Lernmethoden" eingesetzt werden, um die Studierenden dazu zu bringen, selbständig, kreativ und in Teams zusammenzuarbeiten.

4 Unsere Erfahrungen bei der Umsetzung eines Kurskonzeptes basierend auf Open-Source-Software

Die Umsetzung der Lehrveranstaltung „Schlüsselqualifikation 'Digitale Medien' für Lehramtsstudierende"[2], die exemplarisch für diesen Artikel herangezogen werden

2 Universität Bremen, Arbeitsgruppe *DiMeB (Digitale Medien in der Bildung)*, Fachbereich Informatik, in Zusammenarbeit mit Maika Büschenfeldt.

soll, hat die oben erwähnten Ideen in ein modulares Kurskonzept eingebracht. In den Kursen dieser Veranstaltung wurden die Studierenden ohne vorherige Vorbereitung an Computer gesetzt, auf denen *Linux* mit einer KDE-Desktop-Umgebung installiert war.

Aus der Befragung, die vorab durchgeführt wurde, wussten wir, dass alle Teilnehmer bereits gute Kenntnisse des Betriebssystems *Microsoft Windows* hatten und sich mit gängiger Anwendungssoftware (Textverarbeitung, Präsentationsprogramme, Browser) auskannten. Daher entschieden wir uns, den Teilnehmern keine Bedienereinführung in die Grundelemente der Linuxoberfläche und der Programme zu geben. Tatsächlich wurde von den TeilnehmerInnenn nur beklagt, dass man sich zur Lösung einer Aufgabe jeweils für eines von mehreren möglichen Programmen entscheiden musste.

Die Unterschiede zu bereits bekannter Software, die auffällig häufig proprietärer Natur war, waren jedoch kleiner als erwartet: Einige Teilnehmer äußerten, dass sie über *Linux* immer gehört hätten, es sei nur für Techniker geeignet und dass sie nun sehr überrascht seien, dass es sich so einfach und intuitiv handhaben ließe und *Windows* so ähnlich sähe. Besonders gut kam bei allen Teilnehmern an, dass das Programm-Menü übersichtlich strukturiert war. Das erleichterte die Einarbeitung merklich, weil man sich nicht Produktnamen (oder den Hersteller anstatt des Produktnamens) merken musste, sondern in einer entsprechenden Rubrik suchen konnte. Dadurch, dass alles neu war, wurden gerne alle verwandten Anwendungen einer Programmgruppe einmal ausprobiert.

Die Tatsache, dass einige Programme auch für andere Plattformen im Netz zu bekommen sind, führte zu einer leichteren Akzeptanz. Man wusste, dass man ein Programm, das einem gefiel, auch zu Hause installieren und ausprobieren bzw. nutzen können würde, ohne das Betriebssystem zu wechseln. Eine gerne gesehene Eigenschaft betraf das bei unseren Schulungen genutzte *KDE*: Die Teilnehmer schätzten sehr die Vorteile eines Systems, bei dem jeder einzelne Nutzer seine Arbeitsoberfläche in hohem Maße nach eigenen Wünschen einrichten kann.

Ein besonderes Element der Kurse war das Erstellen von Kriterienkatalogen und der Einsatz von *UML (Unified Modelling Language)* als analytischem Werkzeug. Hiermit wurden die Teilnehmer befähigt, ihnen bekannte und auch neue Software zu beurteilen und mit weiterer Software vergleichen zu können. Ebenso konnten sie nach Kursabschluss Wünsche besser darlegen und beschreiben, was für eine eventuelle später gewünschte Mitwirkung in einer Open-Source-Community ein wichtiger Vorteil ist.

5 Erstellung von Kriterienkatalogen

Die Nutzung von Kriterienkatalogen findet sich etwa in technischen Zeitschriften. Sie sollen es dem Verbraucher einfacher machen, eine Kaufentscheidung zu fällen.

Häufig genug sind die Kataloge jedoch lückenhaft oder den eigenen Bedürfnissen nicht genügend gut angepasst. Daher war es unser Ziel, bei den Teilnehmern die Ausbildung einer Kompetenz zu fördern, die es ihnen erlaubt, ihre eigenen Kriterienkataloge zu erstellen. Gleichzeitig wurde trainiert, Softwareeigenschaften kritisch zu analysieren, Software zu bewerten sowie möglichst objektive Entscheidungen dafür oder dagegen zu fällen.

Wie wichtig eine solche Übung ist, zeigt folgendes Beispiel: Studierende testeten zwei Lernsoftware-Produkte und erstellten einen Kriterienkatalog. Obwohl alle Gruppenmitglieder der Meinung waren, dass ein Produkt deutlich besser war, fiel die Bewertung mittels Kriterienkatalogen gleich aus. Das Ergebnis überraschte die Gruppenteilnehmer und machte ihnen deutlich, dass es Eigenschaften gab, die sie nicht durch den Katalog abgedeckt hatten. Bezüglich der Gewichtung der Kriterien gaben viele Kursteilnehmer an, dass ihnen eine leichte Handhabung und gute Zugänglichkeit wichtiger waren als Verfügbarkeit von Source-Code.

6 UML mit Umbrello

Die Werkzeuge der *UML*, häufig als Tool für die objektorientierte Programmierung eingesetzt, umfassen verschiedene Diagrammtypen, die auf verschiedene Aspekte von Problemstellungen zugeschnitten sind. Durch den Einsatz im Kurs soll der kritische Blick auf eine Software, ergänzend zum Kriterienkatalog, um dynamische Aspekte erweitert werden. So werden die Teilnehmer befähigt, die Funktionalität, den Programmablauf und das Zusammenspiel von Softwarekomponenten besser zu verstehen. Verwendet werden die beiden Verhaltensdiagramme Aktivitäts- und Use–Case-Diagramm (Dörge und Büschenfeldt 2006, S. 6).

Anhand eines „Alltagsalgorithmus“ (vgl. Rissberger 1986, S. 6) wird den Teilnehmern erklärt, wie ein alltäglicher Ablauf, z. B. Kaffeekochen, sequentiell dargestellt werden kann. Der Algorithmus wird dann, zusammen mit den Studierenden, einmal als Aktivitäts- und einmal als Use-Case-Diagramm umgesetzt, um ihnen die Nutzung von *UML* zu verdeutlichen. Danach werden die Teilnehmer aufgefordert, sich auf dem Rechner Software auszusuchen, die sie mit den drei Werkzeugen Kriterienkatalog, Use-Case- und Aktivitätsdiagrammen beurteilen sollen. Das Vergleichen der drei Ergebnisse ist der Teil der Aufgabe.

Für die Erstellung verwendeten wir das Open-Source-Programm *Umbrello*, welches die Teilnehmer schnell bedienen konnten. Die gesamte Arbeit wurde schließlich in einem Textverarbeitungsprogramm nach Wahl dokumentiert und in einem Wiki veröffentlicht.

7 Lehrerfolge

Eine Umfrage unter den teilnehmenden Studenten nach Ablauf des Kurses ergab eine signifikante Änderung der persönlichen Einstellung zu Open-Source-Software. Alle Teilnehmer äußerten sich nach den Kursen angenehm überrascht. Besonders erfreulich ist die Tatsache, dass die angehenden Lehrer davon überzeugt waren, dass der Einsatz von Open-Source-Software in der Schule nicht nur sinnvoll, sondern auch im praktischen Sinne geeignet sei. Es stellte sich weiter deutlich heraus, dass auch Anwender eine klare Idee davon haben, was eine Software können sollte und wie die Software am besten einzusetzen sei. Sie haben Ideen, wie sie Unterrichtskonzepte entwerfen können und wie die Software darin Platz finden kann.

8 Fazit

Den Teilnehmern des Kurses wurde die Fähigkeit vermittelt, Software analytisch zu beurteilen und das Ergebnis strukturiert zu dokumentieren. Alle zehn Schlüsselqualifikationen, die Didi et al. nennen, wurden im Kurs umgesetzt:

Kommunikationsfähigkeit Während des Erstellens des Kriterienkataloges mussten die Teilnehmer lernen, sich über Softwareeigenschaften auszutauschen.

Kooperationsfähigkeit Einige Aufgaben konnten nur gelöst werden, wenn gemeinschaftlich gearbeitet wurde.

Flexibilität Unterschiedliche Programme und Tools mussten für die Aufgaben genutzt werden.

Kreativität Entwurf eines Kriterienkataloges und Einsatz eines UML-Tools zur Umsetzung in ein lesbares Diagramm.

Denken in Zusammenhängen Strukturen der untersuchten Software mussten verstanden werden, um die Diagramme erstellen zu können.

Selbständigkeit Einige Aufgaben mussten in Einzelarbeit gelöst werden.

Problemlösefähigkeit Die Dozenten konnten und wollten nicht jede auftauchende Frage klären. Somit mussten einige Probleme selbst gelöst werden.

Transferfähigkeit Früher gemachte Erfahrungen mussten auf die neuen Programme angewendet werden.

Lernbereitschaft Bereitschaft, sich mit neuen Programmen und einem neuen System auseinanderzusetzen.

Durchsetzungsfähigkeit Während der Teamarbeit mussten die Teilnehmer sich mit ihrer Meinung gegen andere Gruppenmitglieder durchsetzen können.

Der Einsatz von Open-Source-Software spielte eine maßgebliche Rolle für die erfolgreiche Vermittlung dieser Qualifikationen.

Literatur

Archan, S. und Tuschek, E. (2002), *Schlüsselqualifikationen – Wie vermittle ich sie Lehrlingen?*, Österreichisches Institut für Bildungsforschung der Wirtschaft, Wien. http://www.ausbildungsjournal.at/art/arc_048_02_schlusselqualifikationen.pdf [02. Jan 2007].

Didi, H.-J., Fray, E., Kloft, C. und Vogt, H. (1993), Einschätzungen von Schlüsselqualifikationen aus psychologischer Perspektive, Zusammenfassung eines Gutachtens im Auftrag des Bundesinstituts für Berufsbildung, Bundesinstitut für Bildungsforschung, Bonn.

Dörge, C. und Büschenfeldt, M. (2006), Vermittlung von IT-Schlüsselkompetenzen für den nachhaltigen Umgang mit Digitalen Medien, *in* 'GdW-Praxishilfen', Luchterhand Verlag, Köln, S. 1–14. Ergänzungslieferung 66, Dezember 2006, Gruppe 9:9.20.20.32.

Gonon, P. (2002), 'Und am Schluss auch noch Latein?', *Panorama* (6), S. 6 ff. http://www.infopartner.ch/periodika/2002/Panorama/Heft_6_2002/pan2606.pdf [02. Jan 2007].

Klafki, W. (1985), *Neue Studien zur Bildungstheorie und Didaktik*, Beltz Verlag, Weinheim, Basel.

Mertens, D. (1974), 'Schlüsselqualifikationen. Thesen zur Schulung für eine moderne Gesellschaft', *Mitteilungen aus der Arbeitsmarkt- und Berufsforschung* **7**, S. 36–43. http://doku.iab.de/mittab/1974/1974_1_MittAB_Mertens.pdf [02. Jan 2007].

Orth, H. (1999), *Schlüsselqualifikationen an deutschen Hochschulen*, Luchterhand, Neuwied.

Rissberger, A. (1986), *Informationstechnische Grundbildung*, Ernst Klett Schulbuchverlag, Stuttgart.

Stephenson, C., Gal-Ezer, J., Haberman, B. und Verno, A. (2005), The New Educational Imperative: Improving High School Computer Science Education, Final Report, CSTA Curriculum Improvement Task Force, New York. http://csta.acm.org/Publications/White_Paper07_06.pdf [02. Jan 2007].

Pädagogische und gesellschaftliche Potenziale freier Software – GNU/Linux in Schule und Elternhaus

HERBERT RECKMANN

(CC-Lizenz siehe Seite 563)

Am Beispiel von GNU/Linux wird die bildungstheoretische und -praktische Relevanz freier Software bewertet: Es wird gefragt, ob durch den Einsatz von Open-Source-Software in der Schule offene curriculare Strukturen begünstigt und die traditionellen Skripte zur Inszenierung von Unterricht innovativ verändert werden können.
Das unterrichtspraktische Potenzial freier Software wird an den konkreten Beispielen der Gestaltung eines Einzelarbeitsplatzes, der Medienecke in einem Klassenraum und des zentralen Server-Systems der Schule analysiert. Dabei wird gezeigt, wie SchülerInnen und Lehrkräfte an ihrem Arbeitsplatz-PC unter einer modernen grafischen Arbeitsoberfläche Programme aus dem Bereich Office, Grafik, Multimedia und Internet nutzen können, um sich in Wort, Bild und Ton auszudrücken. Am Beispiel der Realisierung einer computergestützten Medienecke werden Thin-Clients als robuste, wartungsarme und preiswerte Ausstattungslösung für Lerngruppen, die den Computer relativ frei, unterrichtssituativ ad hoc und eigengesteuert nutzen wollen, vorgestellt. An ausgewählten Fallbeispielen wird bewertet, wie Schulserverlösungen pädagogische wünschenswerte Prozesse der Information, Kommunikation und Kooperationen stützen.

Schlüsselwörter: Bildung · Schule · Unterricht · Medienecke · Schulserver

1 Die neue Ausgangslage

Die Firma *Microsoft* hat für das Frühjahr 2007 *Windows Vista*, das neue Betriebssystem für den Personalcomputer angekündigt und die Preis- und Lizenzbestimmungen seiner Nutzung festgelegt. In zwei Jahren werden Support und Sicherheitsupdating für die in Schulen und Privathaushalten weit verbreiteten älteren Programmversionen

auslaufen. PC-Nutzer stehen spätestens dann vor der Frage, das neue *Windows Vista* zu beschaffen oder mit einem alternativen Betriebssystem zu arbeiten. Längst stehen attraktive Alternativen zu Windows zur Verfügung. Leistungsfähige Betriebssysteme werden nicht nur von kommerziellen Wettbewerbern, z. B. das *OS X* der Firma *Apple*, sondern auch das lizenzkostenfrei angeboten, wie das Programmpaket GNU/Linux.

GNU/Linux ist so genannte Open-Source-Software, die von weltweit kooperierenden frei agierenden Programmierern entwickelt wurde und jedem Interessenten über Internet-Download nahezu kostenfrei zur Verfügung steht. Die freie Software GNU/Linux ist nicht nur als Serversystem, sondern inzwischen auch für den PC–Arbeitsplatz, den Desktop, hervorragend geeignet. Besonders durch die Weiterentwicklung der grafischen Benutzeroberfläche, die Einbindung moderner Multimedia–Funktionalität und hervorragende Hardwareerkennung ist Linux zu einer attraktiven Alternative zu *Microsoft Windows* herangewachsen.

Die Existenz der kostenfreien, funktional gleichwertigen Softwarealternative GNU/Linux sollte die verantwortlichen Akteure des öffentlichen Bildungsbereiches zu neuen Entscheidungen führen. Bundesländer und Gemeinden als Träger des Schulsystems und die Elternhäuser als Finanziers der Lehr- und Lernmittel der SchülerInnen werden genau hinsehen müssen, ob sie die durch den Einsatz von *Vista* verursachten höheren Kosten dauerhaft tragen und veränderte rigide Nutzungsbedingungen akzeptieren wollen.

Im Vergleich zu den Windows-Vorgängerversionen ist *Vista* deutlich teurer und führt zu dauerhaft höheren Belastungen des Bildungsetats für Software. Durch normierende Vorgabe der Softwareentscheidungen in den Schulen würden indirekt auch die Elternhäuser gezwungen, *Vista* zu beschaffen.

Im pädagogischen Kontext ist nicht nur der Kostengesichtspunkt entscheidend. Mit dem Einsatz von *Vista* verändert sich im grundsätzlichen Sinne die Beziehung zwischen Benutzer, Hardware und Software. Haben wir den PC bisher als universell einsetzbare Maschine verstanden, die der Mensch nach eigenem Gutdünken steuern kann, so wird dies mit *Vista* entscheidend verändert. Die Firma *Microsoft* unterwirft Vista-Nutzer mit neuen veränderten Lizenzbestimmungen (*EULA – End User Licence Agreement*) unter ein Regime, das den Charakter der Beziehungen zwischen Hardwarebesitzer und Softwareanwendung in extremer Weise verändert. Zum Beispiel ist es nicht mehr möglich, Hardwarekomponenten zu verändern oder zur Reparatur auszutauschen, ohne das System „reaktivieren" zu müssen. Durch Validierungszwang wird der Einsatz von eigenen Programmen oder Programmen anderer Hersteller erschwert bzw. unmöglich gemacht. Die vom Nutzer nicht verhinderbaren und in Inhalt und Umfang nicht überprüfbaren regelmäßig stattfindenden Datenübermittlungen zerstören den Charakter des PCs als Privat- bzw. Personalcomputer und gefährden grundsätzlich das Prinzip informationeller Selbstbestimmung.

Im Gegensatz dazu bietet Open-Source-Software freie Eingriffsmöglichkeit und vollständige Kontrolle des Nutzers über die Funktionen des PCs. Da der Programm-

code für jeden einsichtig vorliegt und prinzipiell veränderbar ist, wird der Nutzer nicht entmündigt. Im Gegenteil: Die Informationen über Konzept und Mechanik des Programmes sind frei verfügbar, können vermittelt und gelernt werden. Dies ist ein grundsätzlicher Vorteil freier Software und von hoher pädagogischer Relevanz. Der Einsatz von Open-Source-Software im Bereich Schule und Weiterbildung ist nicht nur unter finanziellem Aspekt lohnend, sondern kann sich auch unter pädagogischen und kulturell-gesellschaftlichen Gesichtspunkten als positiv erweisen.

2 Gesellschaftliche und bildungstheoretische Relevanz freier Software

In der folgenden Analyse sollen einige Aspekte des pädagogischen Wertes freier Software herausgestellt und das spezifisch bildungstheoretische Potenzial von GNU/Linux weiter ausgeleuchtet werden.

2.1 Freies Wissen als Basis alltagskultureller Praxis

Die Entwickler von Linux verstehen Open-Source-Software als freies Wissen in alltagsüblicher Selbstverständlichkeit. Menschen haben als Basis ihrer alltagskulturellen Praxis ein gemeinsam geteiltes Wissen erarbeitet; so zum Beispiel Hygienevorschriften zur Erhaltung der Gesundheit, Kochrezepte zur Zubereitung der Speisen, Volkslieder und -Melodien zur Unterhaltung und Anregung. Dies wird als nützliches Basiswissen begriffen, das alle Mitglieder einer Kulturgemeinschaft miteinander teilen und von der älteren Generation der jüngeren übermittelt wird. Dieses gemeinsam geteilte und entwickelte Wissen macht uns lebenstüchtig, ist Wissensbasis, auf die wir uns in Erwerbsarbeit und Freizeit stützen. In diesem Sinne verstehen die Entwickler von GNU/Linux das Wissen *Software*. Wir benötigen es für die Abwicklung unserer alltagskulturellen Belange. Inzwischen stützen wir uns in nahezu allen Lebensbereichen auf computerbasierte Verarbeitungsprozesse von Wissen und Informationen. Der Computer ist Teil der Lebensprozesse geworden, wir sprechen deswegen häufig von der Informationsgesellschaft.

Für das Leben in einer elektronisch digital bestimmten Welt benötigen wir das Wissen Software tagtäglich. Philosophisch betrachtet liegt darin der besondere Wert – die gesellschaftliche und bildungstheoretische Relevanz freier Software. Es ist wichtiges Wissen zur computerbasierten Verarbeitung von Wissen und Information.

Es sollte jedem möglichst frei und ohne Einschränkung zur Verfügung stehen. Es sollte jedem möglich sein, das Wissen weiterzuentwickeln, es an neue Bedingungen anzupassen. Absolute Offenheit und Freiheit zur Veränderung und Verbesserung des Wissens sollten gegeben sein. Nur so könnten die über elektronisch digitale Operationen abgewickelten Lebensprozesse von Reduktionen und Verkürzungen frei gehalten werden. Das freie Gut Software könnte in Gefahr geraten, wenn es der Gemeinschaft entzogen und privatwirtschaftlich angeeignet würde.

Die Auffassung des immateriellen Gutes Software als Ware würde nämlich bedeuten, dass es über Preise limitiert würde, es nicht jedem zugänglich wäre. Auch würde die Verfügung über Strukturierung der Ware beim Eigentümer der Software und somit das Recht und die Macht zur Steuerung der Weiterentwicklung und Veränderung der Software bei ihm liegen.

Mit dem Selbstverständnis freier Software als Teil des Wissens, das im natürlichen Lebensvollzug einer durch elektronisch-digitale Netzwerke bestimmten modernen Welt entwickelt wird, ist die Forderung der Gestaltung des Zusammenlebens nach nicht-pekuniären Kriterien verbunden. Prominente Vertreter der Open-Source-Bewegungen bestehen darauf. Richard Stallman beschreibt dies so:

> „Der fundamentale Akt von Freundschaft unter denkenden Wesen besteht darin, einander etwas beizubringen und Wissen gemeinsam zu nutzen. Dies ist nicht nur ein nützlicher Akt, sondern es hilft, die Bande des guten Willens zu verstärken, die die Grundlage der Gesellschaft bildet und diese von der Wildnis unterscheidet. Dieser gute Wille, die Bereitschaft, unserem Nächsten zu helfen, ist genau das, was die Gesellschaft zusammenhält und was sie lebenswert macht. Jede Politik oder jedes Rechtssystem, das diese Art der Kooperation verurteilt oder verbietet, verseucht die wichtigste Ressource der Gesellschaft. Es ist keine materielle Ressource, aber es ist dennoch eine äußerst wichtige Ressource." (Grassmuck 2002, Klappentext)

Dieses andere Umgehen mit Wissen – das Erhalten einer intakten Infrastruktur zu seiner Entwicklung und seine freie Vermittlung in Erziehung und Bildung – wird auch in der eigenen Arbeitshaltung der Schöpfer freier Software deutlich. In der Analyse von Pekka Himanen (2001) wird nachgezeichnet, was die Entwickler motiviert, in welch' anderem Verständnis von Arbeit und Freizeit sie leben. Der freie Informationsaustausch und die Freude an kreativer Arbeit bezeichnen eine neue Ethik, die Arbeit und Leben „jenseits der Lohngesellschaft" (Gorz 2001) bestimmen könnte, also ein Modell für zukünftige Strukturierung und Steuerung unseres Lebens und der Arbeitsverhältnisse sein kann.

Es ist eine philosophische Leistung der Open-Source-Bewegung, Software als Wissen zu begreifen, das gesellschaftlich bedingt ist, in einer Kulturgemeinschaft entsteht und der alltäglichen Praxis verpflichtet ist. Mit dem Einsatz von Open-Source-Software in den Schulen könnte dies wissenschaftstheoretisch thematisiert werden. Am Beispiel von GNU/Linux könnte verdeutlicht werden, wie Wissen entsteht, wie es im alltäglichen Lebensvollzug getestet und verändert wird und welche – auch hohe ökonomische – Bedeutung das freie Wissen einer Kultur für uns Menschen hat.

2.2 Curriculare und unterrichtsinnovative Relevanz freier Software

Die wissenschaftstheoretische Thematisierung *freien Wissens* ist fundamental und wichtig, jedoch auch sehr abstrakt und abgehoben von den Problemen, die in der Schulpraxis Bedeutung haben. Einen konkreteren Bezug könnte man gewinnen, wenn Analysedimensionen gefunden werden, die der Praxis des Pädagogen näher kommen. Dies können die Fragen der Bestimmung von Unterrichtsinhalten, also der curricular-didaktische Aspekt des Lehrerhandelns und die Probleme der Gestaltung von Unterricht sein.

Eine sehr interessante erziehungswissenschaftliche Analysekategorie, die dem Wissensbegriff sehr nahe kommt, ist das Konzept des „Klassifikations- und Vermittlungsrahmens". Dieser analytische Ansatz wurde von Basil Bernstein (1971) entwickelt. In mehreren Aufsätzen zur Sozialisationstheorie hat er diesen Ansatz entfaltet. Er weist darauf hin, dass vor allem im kontinentaleuropäischen Bereich so genannte *geschlossene Codes* vorliegen, während in den angelsächsischen Ländern, vor allem in den USA, *integrative Codekonzepte* das Unterrichtsgeschehen bestimmen. „Geschlossene Codes" bewirken, dass Unterrichtsinhalte streng fachlich definiert und von anderen Fachdisziplinen abgegrenzt werden. Dies führt zu fachspezifischen Sozialisationsverläufen. Personen, die im Konzept des geschlossenen Codes ausgebildet wurden, bilden dabei unterschiedliche Identitäten aus und tendieren dazu, sich über Fachzugehörigkeit gegeneinander abzugrenzen. Integrative Codes lassen eine freiere Auswahl von Inhalten zu. Typisch dafür ist der Projektunterricht, in dem beispielsweise die Fachgrenzen von Physik, Biologie und Sozialwissenschaft überschritten werden müssen, um ein Problem zu bearbeiten. Hierdurch wird ein fächerübergreifender Lernprozess ermöglicht.

Das Problem, die Frage der Auswahl von Inhalten, wird in der Erziehungswissenschaft auch mit den Fachbegriffen „geschlossene Curricula" versus „offene Curricula" angesprochen. Eng damit verknüpft sind die pädagogisch relevanten Begriffe der *Fremdsteuerung* im Gegensatz zur *Eigensteuerung* im Lernprozess. Eine offene curriculare Konstruktion lässt je nach vorliegendem Problem bzw. Lernsituation unterschiedliche Verfahren der Aneignung zu. Im Gegensatz zum traditionellen Frontalunterricht, der bevorzugten Methode des „geschlossenen Curriculums", wird in offenen curicularen Unterrichtskonstruktionen selbstreguliertes Lernen ermöglicht.

Zur Klärung der Frage, wie der Einsatz freier Software Unterricht verändert, wäre auch die didaktische Kategorie *Unterrichtsskript* bedeutsam. Unter einem Unterrichtsskript versteht man eine ganzheitliche Dramaturgie der Steuerung von Unterricht. Vergleichbarer mit einem Drehbuch, das alle einzelnen Elemente und Bewegungen miteinander verknüpft. In der Skriptkonzeption wird darauf bestanden, dass die Lehrer und Lehrerinnen sich bei der Gestaltung von Unterricht ganzheitlich orientieren. Unterricht ist damit ein komplexer Wechselwirkungszusammenhang, der sich nicht einfach durch Änderung eines einzigen Faktors – zum Beispiel Einführung des Computers und Internet – verändern lässt.

Wählt man diese didaktisch-curriculare Analysekategorie zur Bewertung des pädagogischen Potenzials freier Software, so können relativ konkrete Fragen abgeleitet werden:

1. Verändert der Einsatz freier Software den tradierten Klassifikations- und Vermittlungsrahmen?

2. Führt GNU/Linux in der Schule zu offeneren curricularen Strukturen und zu mehr selbstreguliertem Lernen?

3. Unterscheiden sich die Skripte zur Inszenierung von Unterricht bei jenen Lehrern, die freie Software einsetzen, von jenen, die proprietäre Software benutzen?

Diese Analysedimensionen lassen einen interessanten, differenzierenden Blick auf die Gestaltung von Unterricht zu. Zur Bewertung des Softwareeinsatzes in der Schule sind sie jedoch problematisch, da sie die Existenz und Verbreitung von freier Software und von Linux in der Schule bereits voraussetzen. Es werden entscheidende Faktoren ausgeblendet, nämlich die Prozesse, wie der Personalcomputer und speziell Software als Unterrichtsmittel und Lehr-Lernmedium in die Schule gelangen.

Die meisten LehrerInnen in Deutschland, beispielsweise LehrerInnen aus dem Grundschulbereich, arbeiten mit Windows und haben freie Software wie GNU/Linux noch nicht kennen gelernt. Sie arbeiten weder privat noch in der Schule damit. Wie sollte also die freie Software GNU/Linux in die Schulen kommen? Was bringt Lehrer dazu, Linux einzusetzen? Tatsächlich gibt es inzwischen eine beachtliche Anzahl von „Linux-Schulen" und OSS-nutzende Lehrkräfte. Schaut man genauer hin, wie die Alternativen zur proprietären Software in die Schulen gelangen, so können verschiedene Pfade der Innovation gefunden werden.

Innovationspfad 1: Initiative einer einzelnen Lehrkraft Häufig war es ein Zufallskontakt. In der Anfangsphase der Einführung des PCs und der Internetnutzung in Schulen wurden einzelne Lehrer betraut, den/die Schulcomputer technisch zu warten und die Software zu pflegen. Im Arbeitszusammenhang *Schulen ans Netz e. V.* lernen diese Schulbeauftragten durch Beratung, Fortbildung oder Austausch mit Kollegen Linux-Schulserver-Lösungen (*c't/ODS Arktur*[1]) kennen und mit Linux als Betriebssystem umzugehen. Nicht selten sind es diese Lehrkräfte – meist aus dem mathematischen naturwissenschaftlichen Fachbereich, die Linux in die Schule bringen.

Innovationspfad 2: EDV-Verantwortliche der Kommune In den letzten Jahren wird zunehmend von den Kommunen das für die EDV verantwortliche Personal auch

1 Unter dem Namen *Arktur-Schulserver* wird eine auf GNU/Linux basierende Software angeboten. Diese soll den in den Schulen eingerichteten Computernetzwerken den gemeinsamen Zugang zum Internet erleichtern und die Einrichtung eines Kommunikationssystems ermöglichen. Siehe auch unter http://arktur.schul-netz.de.

dafür eingesetzt, die PC-Ausstattung der Schulen und die Netzanbindung technisch und organisatorisch zu realisieren. Dieses Fachpersonal hat in der IT–Ausbildung UNIX-Know-how erworben und spricht sich auf Grund positiver Erfahrungen mit Stabilität und Sicherheit für Linux als Server-System aus.

Innovationspfad 3: Interessierte Eltern und SchülerInnen In den vergangenen Jahren ist in den populären Medien zunehmend häufig über freie Software, die unter dem Windows-Betriebssystem eingesetzt werden kann, und auch über GNU/Linux als Alternative berichtet worden. Es wurden Linux-Distributionen für den heimischen Desktop vorgestellt, die sich durch eine nutzerfreundliche grafische Oberfläche, gute Multimedia-Fähigkeit und leichte Installierbarkeit auszeichnen. Interessierte Eltern haben diese Angebote aufgegriffen und Open-Source-Programme genutzt. Sie werben für den Einsatz freier Software in den Schulen. SchülerInnen oder Eltern, die Linux privat für den Schreibtisch-PC nutzen, sprechen sich häufig auch für den Einsatz von Open-Source-Software in der Schule aus.

Aus diesen Beschreibungen ist ablesbar, dass die Einführung von Linux in den Schulen bisher sehr zufällig und unsystematisch verlief. Es kann kein Zusammenhang gefunden werden zwischen pädagogischer Orientierung der Schulen und Präferenz freier Software. Es könnte sein, dass freie Software an solchen Schulen leichter eingeführt und genutzt werden kann, wo wir schon Tendenzen der fortschrittlichen Unterrichtsinszenierung erleben, wo offenere curriculare Strukturen und moderne Unterrichtsskripte das Geschehen bestimmen.

Wie wir es bei der Einführung von Computer und Internet schon beobachten konnten, werden in den Schulen, in denen konservative unterrichtliche Konzepte vorherrschen, trotz des Einsatzes neuer Technologien die traditionellen Verfahren beibehalten, so z. B. lehrerzentrierter Frontalunterricht in modernen Computerkabinetten. Neue Technologien führen nicht automatisch zu Veränderung von Schule. Das wird bei der Einführung von freier Software und Linux nicht anders sein. Wir können die Vermutungen äußern, dass das innovative Potenzial freier Software und Linux am ehesten dort genutzt wird, wo bereits eine offene pädagogische Orientierung besteht, also eine Tendenz zu integrativen Vermittlungcodes, zu offener curricularer Gestaltung und wo Unterricht schon heute durch fortschrittliche Unterrichtsskripte bestimmt ist.

2.3 Freierer Zugang zu Lern- und Lehrressourcen

Das pädagogische Potenzial freier Software ist also im didaktisch-curricularen Bereich nicht sicher bestimmbar. Eindeutiger sind die Folgewirkungen des Einsatzes von Open Source jedoch in einer anderen pädagogisch wichtigen Dimension, nämlich in der Kostenfrage. Mit freier Software können die finanziellen Belastungen, die als Barriere für

den Zugang zu Lehr- und Lernressourcen wirken, für Elternhaus und Schule deutlich verringert werden. Im vorigen Jahrhundert war mit der Einrichtung öffentlicher Schulen und der gesetzlichen Schulpflicht die allgemeine Lernmittelfreiheit verbunden worden, um SchülerInnen unabhängig vom Finanzstatus der Herkunftsfamilie das Lernen zu ermöglichen. Lernmittelfreiheit ist ein Element zur Sicherstellung gleicher Bildungschancen. Open-Source-Software ist solch ein freies Lernmittel. Insbesondere für SchülerInnen aus weniger wohlhabenden Familien, den sozialen Unterschichten, kann freie Software ein willkommenes Lehr- und Lernmittel sein und den Erwerb von Medienkompetenz und die Ausbildung von *computer literacy* erleichtern.

Kostenersparnisse durch den Einsatz freier Software sind nicht nur für den PC-Einzelplatz des Schülers und des Lehrers zu erwarten, sondern auch bei der Einrichtung von Schul-Netzwerken. Auf der Basis freier Software kann die Netzwerkinfrastruktur von Schulen kostengünstig ausgebaut werden. Partizipation und aktive Mitbeteiligung wird ermöglicht, insbesondere durch Nutzung der freien, nicht kostenpflichtigen Webangebote wie beispielsweise *Wikipedia* und *OpenWebSchool*[2].

Sollte dennoch spezielle Anwendungssoftware für den schulischen pädagogischen Einsatz noch nicht entwickelt sein, so ist es möglich, die Ausarbeitung schulpädagogisch relevanter Software anzuregen. Innerhalb der Linux-Entwicklergemeinden gibt es Gruppen und Arbeitsansätze, die auf das Einsatzfeld Schule, Unterricht und Edutainment ausgerichtet sind.

Die Anregung und Unterstützung zur Entwicklung pädagogisch relevanter Software sollte auch stärker von der öffentlichen Hand kommen. Es ist dringend erforderlich, dass die verantwortlichen Entscheider des öffentlichen Schulsystems sich für die Entwicklungen freier Software engagieren. Eine bessere Kommunikation und Abstimmung zwischen Akteuren im Arbeitsbereich Softwareentwicklung und denen im Bildungsbereich, Erziehung und Elternhaus könnte die Entwicklung pädagogisch relevanter freier Software entscheidend beschleunigen.

3 Unterrichtspraktisches Potenzial freier Software

Wie bereits dargestellt wurde, kann Linux als nahezu kostenfreies Lern- und Lehrmittel für SchülerInnen und Lehrkraft bewertet werden. Durch die immensen Fortschritte in den letzten Jahren ist Linux inzwischen auch für den Einzelarbeitsplatz, für den Schreibtisch-PC, Betriebssystem der ersten Wahl geworden.

2 Die *OpenWebSchool* (http://www.openwebschool.de) bietet Unterrichtseinheiten an, die aus dem Internet heraus benutzt werden können. Diese Unterrichtseinheiten können einfach in einem Browser aufgerufen werden und sind somit ohne jegliche Installation und Betreuung auf jedem Internet-fähigen Rechner sofort einsetzbar.

3.1 Linux für den Schreibtisch-PC

Mit GNU/Linux hat jede/r die Möglichkeit, sich auszudrücken in Wort, Bild und Ton und über das weltweite Netz zu kommunizieren. Die pädagogisch wichtigsten Funktionen sind:

Sich informieren Es ist sehr leicht möglich, sich an einem Linux-Arbeitsplatz über das Internet Informationen zu beschaffen. Zum Beispiel können über Web-Recherchen oder durch Nutzung von Angeboten spezieller Portale, sei es der Schulserver oder der Bildungsserver einer Region, Informationen eingeholt werden.

Mit anderen kommunizieren An einem Linux-Arbeitsplatz kann man alle Kommunikationstechniken nutzen, die heutzutage die neue Informationstechnologie bietet. So kann man Dateien bearbeiten, versenden und sie in Kommunikation mit anderen gestalten oder sich über E-Mail mit anderen austauschen oder an einem Online-Präsenzsystem wie beispielsweise Internet-Chat teilnehmen.

Mit anderen kooperieren Auch für Bereiche des gemeinsamen Arbeitens stellt Linux einen umfangreichen Satz von Anwendungsprogrammen zur Verfügung. So ist es möglich, über Internettechnik Mitglied in einer so genannten geschlossenen Benutzergruppe zu werden und mit den anderen Teilnehmern in einem virtuellen Raum zu arbeiten. Es können Dateien auf dem Arbeitsraumserver abgelegt werden, um sie anderen Arbeitsgruppenmitgliedern zur Durchsicht und Überarbeitung zur Verfügung zu stellen. Es lassen sich Online-Treffen organisieren, so dass SchülerInnen sich von zuhause aus an einer Gruppendiskussion beteiligen können. Natürlich ist es auch möglich, dass SchülerInnen und LehrerInnen miteinander auf diesem Wege kommunizieren und kooperieren.

Die Funktionsbreite von GNU/Linux speziell für den pädagogischen Kontext kann sehr gut am konkreten Beispiel *Seminarix*[3] gezeigt werden. Das *Studienseminar Neuss* nutzt bereits das alltagspraktische Potenzial freier Software. Dort wurde für die Lehramtskandidaten der Sekundarstufe eine Sammlung pädagogisch relevanter freier Anwendungssoftware auf CD-ROM zusammengestellt. Auf dieser direkt bootbaren Linux-Live-CD-ROM (Basis *Debian Linux*, Distribution *Kubuntu*) finden Lehramtkandidaten, Schülerinnen, Schüler und Lehrkräfte eine nach Schulfächern kategorisierte Sammlung leistungsfähiger Programme.

So zum Beispiel existiert für den Office-Bereich die hervorragend geeignete *OpenOffice.org Suite*, die Textverarbeitung, Präsentationsgrafik, Tabellenkalkulationen, Zeichnen- und Malprogramme, Termin- und Kalenderfunktionen zur Verfügung stellt. Als Programme zur Grafik-/Bildbearbeitung werden z. B. das Bildbearbeitungsprogramm *GIMP* und die Scannersoftware *SANE* mitgeliefert. Auch für die Multimedia-Wiedergabe und -Bearbeitung wird freie Software angeboten. Zum Beispiel

3 Informationen zum Projekt *Seminarix – die Live-CD für die Lehrerausbildung* finden sich unter http://www.seminarix.org.

zur Bearbeitung von Audiodaten, Musik auf der Basis von MP3- oder OggVorbis-Kompressionsverfahren oder von audiovisuellem Material wie Film, Video und Digitalfotografie.

Natürlich stellt der Linux-Desktop unzählige Programme für die Arbeit im Internet bereit, sei es vom einfachen Datentransfer, über Browserprogramme bis zu komplexen Content-Management-Systemen und E-Learning-Tools. GNU/Linux ist ja ein Kind des Internets und für die Arbeit im digitalen elektronischen Netzwerk bestens geeignet.

Dies zeigt, dass es möglich ist, durch schulspezifische Zusammenstellungen von Programmen auf CD-ROM einen von SchülerInnen wie auch von Lehrkräften direkt nutzbaren gemeinsamen Programmpool kostenfrei zur Verfügung zu stellen. So können sich SchülerInnen und Lehrkräfte bequem auf diese Sammlung stützen und sich im Unterricht auf die wesentlichen inhaltlichen Aspekte konzentrieren.

3.2 Medienecke

Die dargestellten Vorzüge des Linux-Desktops lassen sich nicht nur in der Arbeitssituation am Einzelarbeitsplatz nutzen, sondern pädagogisch spezifischer auch in der Schulklasse bzw. im Unterrichtskontext. In der Regel zielt die Inszenierung von Unterricht auf größere Lerngruppen von 15–30 Personen, zum Beispiel auf Klassenverbände in der Form der altershomogenen Jahrgangsgruppen. Neuere Entwicklungen der Unterrichtsgestaltung, in der beispielsweise die Jahrgangsklasse aufgelöst ist und freie Assoziationsformen von Lernenden zugelassen werden, so genannte leistungs- und altersheterogene Gruppen, sind im öffentlichen Schulwesen des deutschsprachigen Raumes noch nicht sehr verbreitet.

In der Schulpraxis finden wir unterschiedliche Ausprägungen der pädagogischen Arbeit mit Schulklassen. In vielen Schulen wird noch im traditionellen Verfahren des Frontalunterrichtes gearbeitet. Hier wird der Einsatz des Computers häufig in der Form so genannter Computerkabinette organisiert. Das heißt konkret, dass jeder Schüler in der Schulklasse vor einem PC sitzt und die Lehrkraft frontal unterrichtet.

Eine andere Version – glücklicherweise inzwischen die verbreitetste – ist die, in der die SchülerInnen einer Schulklasse differenziert arbeiten. Je nach spezifischer Vorbedingung des Schülers – nach Lernstand – werden Arbeitsaufträge erteilt oder Lernmöglichkeiten angeboten. In diesen Fällen individualisierten Unterrichts wird der Computer als Teil der Lernmedien-Ausstattung einer Schulklasse gesehen. Es wird im Klassenraum eine so genannte Medienecke eingerichtet, in der der Schüler bei Bedarf Ressourcen zum Lernen findet, beispielsweise Lexika, Lehrbücher, Mikroskop, Tonband, Kopfhörer und Videorecorder.

In solchen Medienecken ist der PC mit seinen peripheren Geräten wie Drucker, Scanner und Digitalkamera Teil der technischen Ausstattung. So können SchülerInnen je nach ihren eigenen Lernbedürfnissen oder je nach Arbeitsvorgabe des Lehrers frei diese Ressourcen nutzen. In Medienecken sind zwei bis sechs PC-Arbeitsplätze durchaus ausreichend, um pädagogisch optimal agieren zu können.

Die Medienecke ist eine unter pädagogischen Gesichtspunkten gestaltete Gelegenheitsstruktur. Für solche Kontexte hat sich ein besonderer Ansatz des Einsatzes von freier Software bewährt, der den technischen Wartungsaufwand für PCs extrem reduziert. Es ist das Konzept des so genannten *Linux Terminal Server Project (LTSP)*[4]. Der Verein *Freie Software und Bildung e. V.*[5] hat vor kurzem eine solche Terminalserver-/Thin-Client-Lösung erarbeiten lassen und stellt sie über den Entwickler Martin Herweg zur Verfügung. Diese Linux-Live-CD-ROM heißt *EduKnoppix*[6]. Es ist, wie der Name sagt, eine modifizierte Version der bekannten Linux-Distribution *Knoppix*. Also eine bootbare CD-ROM, von der die Open-Source-Programme gestartet werden können. Diese Linux-Version lässt sich auch auf Festplatte installieren und bietet dann die volle Funktionalität eines Terminalservers.

Je nach technischer Ausstattung des Servers können mehrere (bis zu 20) Clients bedient werden. Dieser Ansatz hat sich bestens bewährt. Beispielsweise kann eine solche Konstruktion in der offenen Jugendarbeit genutzt werden oder für die Surfstationen eines Internetcafés. Da die Thin-Client-Rechner plattenlos arbeiten und über *Etherboot* gestartet werden, sind sie ausgesprochen robust und gegen Vandalismus geschützt.

Werden in einer Medienecke mehrere Thin-Clients an einen leistungsfähigen Terminalserver angeschlossen, kann der größte Teil der im vorherigen Kapitel zur Desktop–Situation beschriebenen Software genutzt werden. Besonders hervorzuheben ist, dass sich jeder Anwender seine spezifische Benutzeroberfläche einstellen kann. Zum Beispiel können Spracheinstellungen muttersprachlich vorgenommen werden. Dies ist in Schulklassen mit hohem Migrantenanteil – pädagogisch bewertet – ein großer Vorteil.

3.3 Linux im Schulnetzwerk

Um die pädagogische Relevanz freier Software zu analysieren, reicht es nicht aus, die Einsatzbereiche Desktop/Einzelarbeitsplatz oder PCs in der Medienecke eines Klassenraumes zu analysieren. Wichtig ist auch der Blick auf das organisatorische Gesamtsystem der Einzelschule. Die Perspektive kann noch erweitert werden auf die Schulen in einer Stadt oder einer Gebietskörperschaft. Auch auf dieser Ebene müssen für die Schulen Dienstleistungen und Funktionen zur Verfügung gestellt werden. Es bietet sich an, den EDV-Service in solch komplexen Gebilden netzwerkartig zu organisieren. Über zentrale Server können Leistungen erbracht und Funktionen abgewickelt werden.

Wenn wir uns umschauen, wie die einzelnen Schulen oder Kommunen ihre informationstechnischen Belange regeln, sehen wir eine große Spannweite verschiedenster Realisationen. Es gibt Kommunen, die eine mit den einzelnen Schulen abgestimmte Gesamtnetzwerkkonstruktion erarbeitet haben, ein integriertes Schulnetzwerk auf

4 http://www.ltsp.org
5 http://fsub.schule.de
6 http://media.lug-marl.de/doc/eduKnoppix2/

kommunaler Basis etabliert haben. Wiederum findet man in anderen Gebietskörperschaften nur isolierte Schullösungen. Hier hat die einzelne Schule für sich ein Netzwerk errichtet und die PCs miteinander verbunden. Sehr häufig findet man noch – vor allem bei kleinen Schuleinheiten im Grundschulbereich – Einzel-PC-Lösungen. Die Rechner sind nicht miteinander vernetzt, der Zugang zum Internet wird über ISDN- oder DSL-Router gewährleistet.

Die Bandbreite der Lösungen lässt sich etwa folgendermaßen systematisieren:

- Ansammlung von Schreibtisch-PCs, z. B. Internetanschluss über ISDN oder DSL-Router/Switch
- Kommunikationsserver-Lösungen, z. B. *c't/ODS-Schulserver Atktur*
- Linux-Terminalserver-/Thin-Client-Lösungen, Beispiele sind *EduKnoppix* oder *Edubuntu*
- Schulserver-Lösungen, Beispiele: Musterlösung Baden-Württemberg, *Skolelinux*

Dies sind technische Lösungen, die verschiedene Chancen zur Verwirklichung pädagogischer und gesellschaftlicher Ziele bieten. Wie bereits zum Schreibtisch-PC und zur Terminalserverlösung ausgeführt wurde, bieten Linux-Lösungen bereits auf den unteren Stufen netzwerktechnischer Aggregation schulpraktisch und unterrichtsbezogen große Vorteile. In der Konstruktion schulweiter oder kommunaler Gesamtlösungen liegen jedoch weit größere Potenziale.

Netzwerklösungen eröffnen Zusatzleistungen, die die Verwirklichung pädagogischer Intentionen unterstützen können. Die wichtigen Funktionen Information, Kommunikation, Kooperation können über Netzwerklösungen stabiler und vollständiger realisiert werden. Vor allem lassen Netzwerklösungen es zu, dass sich die Akteure in der Schule – also SchülerInnen, die Lehrkräfte und die erziehungsberechtigten Eltern – leichter aufeinander beziehen können und somit bei Bedarf lehr- und lernrelevant kommunizieren können. Es ist möglich, innerhalb der Schulgemeinschaft und auch auf kommunaler Ebene über Einzelschulen hinausgehend eine Internetplattform zu schaffen, so genannte virtuelle Räume, in denen sich die Akteure assoziieren und austauschen können. Diese virtuellen Räume ermöglichen es, dass Schüler, Schülerinnen und Lehrkräfte miteinander kommunizieren und somit am schulischen und kommunalen Miteinanderleben partizipieren.

4 Zusammenfassung und Ausblick

Wird in einer Schule oder in einer Kommune eine Linux-Netzwerklösung eingeführt und dies offen kommuniziert – also curricular thematisiert, so erhöht sich damit die Chance, die besonderen Vorteile freier Software für die Ausbildung einer freien Wissenskultur deutlich zu machen. Freie Software ist freies Lernmittel und auch für

Kinder aus weniger wohlhabenden Familien leicht zugänglich. Dies kann die Ausbildung von *computer literacy* fördern und die allgemeine Medienkompetenz stärken. Es können medienerzieherisch wichtige Lernziele erreicht werden, wie zum Beispiel die Sensibilisierung für informationelle Selbstbestimmung, freie Meinungsäußerung gegen ungerechtfertigte Netzkontrolle. Aspekte der Netzwerksicherheit können auf einer Basis diskutiert werden, in der der Einzelne Mitbeteiligung findet.

Freie Software kann problemlos vervielfältigt, verbreitet und getauscht werden. Damit können die natürlichen menschlichen Leistungen des ungehinderten Austausches von Wissen gestützt, die Kultur des Schenkens und Teilens gepflegt und somit gesellschaftlich wertvolle Orientierungen entwickelt werden.

Literatur

Bernstein, B. (1971), 'Klassifikation und Vermittlungsrahmen im schulischen Lernprozess', *Zeitschrift für Pädagogik* **17**(2), S. 145–173.

Gorz, A. (2001), Welches Wissen? Welche Gesellschaft?, *in* 'Gut zu Wissen', Heinrich Böll Stiftung. http://www.wissensgesellschaft.org/themen/orientierung/welchegesellschaft.html [05. Jan 2007].

Grassmuck, V. (2002), *Freie Software – Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn. http://freie-software.bpb.de [05. Jan 2007].

Himanen, P. (2001), *Die Hacker-Ethik und der Geist des Informations-Zeitalters*, Riemann Verlag, München.

Zwei IT-Welten für alle SchülerInnen – Schule und freie Software

CHRISTIAN DORNINGER

(CC-Lizenz siehe Seite 563)

In diesem Artikel wird die Open-Source-Szene an Österreichs Schulen skizziert. Einige Zahlen zu den Einsatzbereichen von Linux zeigen positive Entwicklungen seit 2003 an, die „Zwei-Plattformen-Strategie" für die Schulen wird vorgestellt und die seit 2005 nachhaltige Linux-Initiative führt zu einer neuen Sicht kollaborativer Lernprozesse im Bildungswesen. An den Schulen erhofft man sich durch den Einsatz von Open-Source-Software mehr Flexibilität und Vielfalt in der Ausbildung und mehr Rechtssicherheit durch eine Nutzung der *General Public License* bei Schülern und Studenten.

Schlüsselwörter: Bildung · Schulen · IT-Strategie · Lernmanagementsysteme

1 Skepsis am Beginn

Dunkel in Erinnerung blieb eine Art Anfang im Dezember 2002: Einige Lehrer hatten sich in den Kopf gesetzt, ein österreichisches Open-Source-Zertifikat mit einem Installationsworkshop aus der Taufe zu heben. Es war eine Art produktives Chaos in der Vorweihnachtszeit – viele Tische im Forum einer neuen technischen Schule im Wiener 3. Bezirk, es war durchaus eine Installationsparty für Schüler und wenige Schülerinnen, es gab Vorträge zu Linux, große Open-Source-Gefühle und wenig Moderation.

Wir planten Auftritte bei IT-Jour-Fixe-Veranstaltungen, Sommerseminare mit Open-Source-Produkten „in schönen Gegenden" in den Alpen, die Teilnahme an eingeführten herbstlichen Fixpunkten wie dem Vorarlberger Linux-Tag in Dornbirn und immer wieder einen Abschluss so um den 5. Dezember – „Krampustag ist Linux-Tag" hat sich langsam eingebürgert.

Im frühsommerlichen Wien gab und gibt es im lauschigen Museumsquartier dann die Wiener Linux-Wochen – ein Treffen von freundlichen Leuten, vielen Medienarbei-

tern, die in freien Gruppen die Ideale der *General Public License* predigten und gute Anwendungen vorzeigten. Ein bisschen nach Insiderslang klangen die Vorstellungen der Projekte; man brauchte einiges an Vorwissen, um wirklich mithalten zu können.

Für einen in der täglichen Projektarbeit stehenden Fachbeamten aus dem IT-Bereich gab es eine Menge positiver Signale, viele Jungen, die wirklich gut waren, Schüler aus unseren technischen Schulen, die plötzlich als eine Art Szeneguru erkannt wurden oder auch länderübergreifende Kontakte, die sich bis heute gehalten haben. Und das alles praktisch ohne finanzielle Mittel im Hintergrund: Es sei eine Ehre, bei Linux-Meetings antreten zu können, Vortragshonorar wolle man dafür keines, so die oftmalige Antwort beim Anheuern von Vortragenden für neue Events.

Aber es gab auch Irritationen: Vieles sah nicht wirklich durchorganisiert aus. Auf Anfänger wurde mit dem „Insiderkauderwelsch" keine Rücksicht genommen, im Endeffekt wollte niemand, auch nicht in der Lehrergruppe, eine gediegene Veranstaltungsplanung vorantreiben. Firmenvertreter von großen IT-Konzernen entpuppten sich bald als reine Werbeträger – oder ließen das „gemeinsame Projekt" einfach im Stich. Wohin man blickte, sah man intelligente Individualisten mit sehr viel Ahnung über zukünftige IT-Anwendungsbereiche. Wo bleibt da die Verbindlichkeit, die man von anderen Tätigkeiten gewöhnt ist?

Die starke Metapher von Raymond (2001), bei der Softwareentwicklung zwischen „Kathedrale" oder „Basar" zu unterscheiden und die Möglichkeiten des Basars als subsidiäre Steuerung zu entdecken, hat viele neue Erkenntnisse für den Softwareeinsatz im Bildungsbereich gebracht und so um 2003 Entwicklungen vorweggenommen, die wir im pädagogischen Geschäft erst jetzt langsam zu ahnen beginnen. Mit diesem Beitrag soll versucht werden, die Qualitäten und Quantitäten von Open-Source-Projekten im österreichischen Schulwesen kurz zu beschreiben und – eher ungewöhnlich für ein eher technisches Thema – Erkenntnisse für eine neue Lernkultur abzuleiten.

2 Einige Zahlen

Der Begriff *Open Source* wurde mit der Offenheit des Quellcodes, also der Möglichkeit, Einblick in den Aufbau und die Funktionsweise von Programmen zu erhalten, identifiziert. Bildungsbetriebe und SchülerInnen bzw. Studierende interessiert natürlich die Nutzung der Freiheiten, die durch die *General Public License* gewährt werden, besonders. Daher erscheint heute *free software* oder „freie Software" passender. Freie Software setzt sich an Österreichs Schulen langsam durch. Zwei zur Verfügung stehende Datenquellen über die Einsatzmöglichkeiten zeigen in dieselbe Richtung: Zur Erstellung von Mengengerüsten für Vertragsverhandlungen zu Softwareprodukten werden zweijährlich gestartete Online-Befragungen von Schulstandorten durchgeführt, in denen erfragt wird, welche Betriebssysteme auf ihren Schul-PCs laufen (IT-Lenkungsgruppe im Bundesministerium für Bildung, Wissenschaft und Kultur 2005). Die Auswertungen der Jahre 2003 und 2005 weisen einen Auswertegrad von

Schultyp	Linux 2003	Linux 2005	WinX 2003	WinX 2005
HTLs	1 266	1 677	12 345	12 436
Berufsschulen[1]	11	169	5 405	4 776
AHS	847	1 540	12 191	16 848
Hauptschulen	270	520	15 031	23 788
Polytechnische Schulen	105	231	4 359	4 420
HAK/HLW	267	520	17 245	20 744
LMS Österreich[2]		163 (Moodle)		49 (Class-Server)

Tabelle 1: Vergleich der an Schul-PCs installierten PC-Betriebssysteme in den Jahren 2003 und 2005 (Linux 200x meint freie Software und WinX 200x Microsoft Windows). Viele der Installationen sind parallel auf den Schulservern installiert, wobei die Mehrzahl der Schulen Windows als hauptsächliches Betriebssystem führt. Der Aufholprozess von Linux ist jedoch deutlich erkennbar.

ca. 65 Prozent auf (die Zahlen werden dann hochgerechnet) und ergeben das in Tabelle 1 dargestellte Bild.

Die in der Tabelle 1 genannten Schultypen sind Teil der Sekundarstufe des österreichischen Schulwesens:

- Hauptschule: 10- bis 14-Jährige
- Allgemeinbildende Höhere Schulen (AHS): 10- bis 18-Jährige
- Höhere Technische Lehranstalten (HTLs): berufsbildende Vollzeitschulen für 14- bis 19-Jährige
- Berufsschulen in dualer Form: 15- bis 18-Jährige
- Polytechnische Schule: 14- bis 15-Jährige
- Handelsakademien, Höhere Schulen für wirtschaftliche Berufe (HAK/HLW): berufsbildende Vollzeitschulen für 14- bis 19-Jährige

Die Steigerung bei Linux und freier Anwendersoftware ist festzustellen; „Hochburgen" sind technische Schulen, allgemeinbildende Schulen und Hauptschulen in manchen Bundesländern (z. B. Tirol und Vorarlberg). Das Verhältnis zu Windows ist noch immer 1:10 zu Ungunsten der freien Software. Viele Open-Source-Installationen sind Zweitinstallationen zum „Ausprobieren". Bei den Lernplattformen zeigt sich aber sofort ein anderes Bild, auf das noch zurückgegriffen wird.

1 Die Erhebung an den Berufsschulen ist unvollständig, zeigt aber eine Tendenz.
2 „LMS in Österreich" bedeutet Lernmanagementsysteme an allen Schulformen.

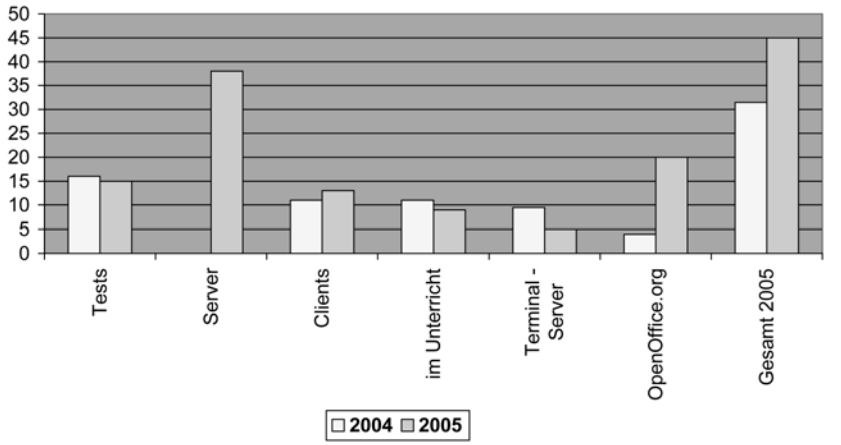


Abbildung 1: Einsatzbereiche von Linux (Erfahrungen 2004/2005)

Eine ausführliche Untersuchung zu „Linux an den Schulen" führte die Donau-Universität Krems im Januar/Februar 2004 und Februar/März 2005 durch (Schwed 2005): Im ersten Durchgang wurden 640, im zweiten 437 Schulen befragt und Linux-Distributionen im Labor unter typischen Schulbedingungen getestet (siehe Abbildungen 1 und 2).

Festzuhalten ist der Anstieg der Linux-Zweitinstallationen (immerhin 21 % der von Krems befragten Schulen gaben eine Zweitinstallation an, aber nur 2 % nutzten sie mehrheitlich und nur weitere 2 % nutzten Linux hauptsächlich). Bei *OpenOffice.org* sind die Zahlen etwas besser. Terminal-Server-Lösungen sind trotz des effizienten Geräteeinsatzes wenig geläufig. Bei der Evaluierung von Distributionen für den Schuleinsatz (EDV-Raum mit 15 PCs) wurden die Distributionen *OpenSchoolServer* und *Sphinx* mit einem „sehr gut" bewertet.

3 Die Zwei-Plattformen-Strategie

Eine gleitende Einführung von Linux als Betriebssystem war uns bis Mitte 2003 nicht ganz klar: War eine Substitutionsstrategie angebracht oder sollte man zuerst Erfahrungen sammeln? Klaus Knopper, Softwareentwickler aus Kaiserslautern, zum Sommerseminar nach Tamsweg im Bundesland Salzburg eingeladen, brachte eine gute Idee mit: Seine mit Studenten und Kollegen erstellte Edition *Knoppix* (damals Version 2.4) ist eine von der CD lauffähige Zusammenstellung von Softwareprodukten unter der *GNU General Public License* mit automatischer Hardwareerkennung und Unterstützung für Grafikkarten, Soundkarten, SCSI- und USB-Geräte. *Knoppix* kann

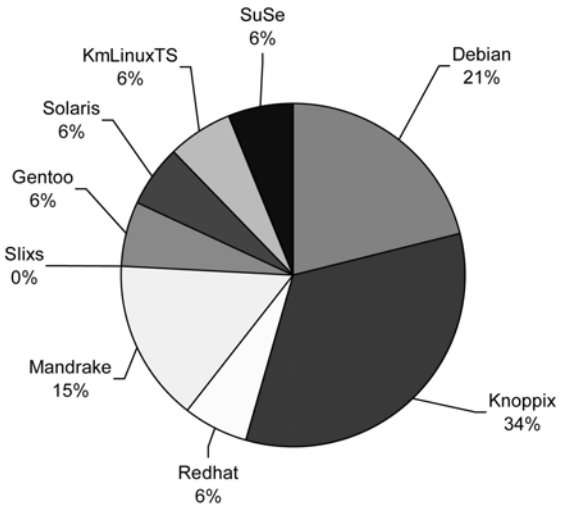


Abbildung 2: Distributionen im Clientbereich

als Linux-Demo, Schulungs-CD, Rescue-System oder allgemein als Plattform für Anwendersoftware angepasst und eingesetzt werden. Es ist keinerlei Installation auf der Festplatte notwendig. Alles in allem ideal für Leute, die ein alternatives Betriebssystem vorsichtig probieren wollen.

Besonders in unserem kaufmännischen Schulwesen (Handelsakademien = HAKs), wo die Notwendigkeit des Arbeitens mit den am Markt führenden Produkten immer wieder argumentiert wird, erschien eine Testversion dieser Alternative besonders nützlich. Der Schuldirektor der HAK Tamsweg bemühte sich um eine Verteilung dieser eduKnoppix-CD. Die Aktivität hat sicher zum Ansteigen der Testkonfigurationen an den weiterführenden Schulen beigetragen. Ein Sommerseminar wurde seit 2003 mit großem Erfolg durchgeführt.

2004 war die Zeit reif für eine größere Veranstaltung, die unter dem Titel „Linux für die Schule“ am 20. September 2004 an der Donau-Universität Krems durchgeführt wurde. Die Kremser bemühten sich um ein professionelles Ambiente, viele Lehrende und EDV-Kustoden waren anwesend, um die ersten Zwischenergebnisse der oben genannten Studie (Schwed 2005) zu diskutieren. Damals glückte den Veranstaltern der Donau-Universität und des Bildungsministeriums auch ein Schlagwort, das gut in den Medien untergebracht wurde: „Schülerinnen und Schüler der Sekundarstufe II sollen sich in zwei EDV-Welten zurechtfinden, eine davon im Open-Source-Bereich“. Hingewiesen wurde unter anderem auf die zuverlässigen Produkte im Server-Bereich,

die Resistenz gegen Viren und die ständige Weiterentwicklung der begonnenen Versionen. Natürlich zählt auch der Kostenfaktor, wenn auch an zweiter Stelle. Chris DiBona, Programm-Manager bei *Google*, gab einen mitreißenden Ausblick auf die *big years* für Linux.

Bei der Veranstaltung versuchte man auch, mit dem deutschsprachigen Wort „freie Software" etwas von den anderen Akronymen abzurücken. Die zweite Veranstaltung im Oktober 2005 nannte sich allerdings wieder „Open Source Gipfel" – zu verführerisch war der eingespielte Begriff. Dort gings ins Praktische: Musterlösungen für zentrale Angebote wurden dargestellt, Open Content und *Creative Commons* waren ein Thema und die Studie aus Krems wurde in ihrer Endform präsentiert.

Die Zwei-Plattformen-Strategie sollte nun mit Leben erfüllt werden: Im Sommer 2004 unterzeichnete die Bildungsministerin ein Abkommen mit *SUN*, um *StarOffice* in der Version 7 allen Schulen gratis zur Verfügung zu stellen (das Abkommen schloss eine Hotline und ein Mindestmaß an Betreuung mit ein). Die in Österreich recht beliebte und prinzipiell produktunabhängige *European Computer Driving License (ECDL)* wurde nun auch mit einer E-Learning-CD der Fa. *Bitmedia* angeboten, die sich auf Open-Source-Produkte bezog (*Linux JDS/Fedora Core*, *StarOffice*, *OpenOffice.org*, *Mozilla* u. a.). Im Endeffekt wurden die LAMP-Arbeitsumgebungen (Linux, *Apache*, *MySQL* und *PHP*) an den österreichischen Schulen deutlicher sichtbar gemacht. Nicht zuletzt läuft auch eine im berufsbildenden Schulwesen viel verwendete Schülerverwaltungssoftware (genannt *SAS-II*, *Schüler-Administrations-System*) der EDV-Versuchsanstalt der technischen Lehranstalt in St. Pölten als Open-Source-System auf einem Tomcat-Server.

Dies drückte sich im Herbst 2004 schlussendlich auch in der Gesamtstrategie für den Bildungsbereich in Österreich aus. Ähnlich der Regelung für die Softwareentwicklung in Dänemark[3] definierte man in Österreich Kriterien für den Softwarekauf an Schulen und Universitäten (Menzel und Psihoda 2004).

Das in Österreich geläufige Entwicklungsprogramm heißt „eFit" (*electronic fit*) und wird seit 2000 gefahren. 2004 folgte mit dem Begriff „eFit2" eine zweite Auflage, die nun auch eine Software-Ankaufs- und Einsatzstrategie enthält. Zwei Passagen daraus:

> „Eine Balance zwischen kommerziell vertriebenen Softwareprodukten und Open-Source-Lösungen wird angestrebt [. . .]. Eine Forcierung von Open-Source-Community-Projekten, die einerseits gute Lösungen auf Basis der *General Public License* ermöglichen, aber andererseits auch engagierten Jungen jene projektartige Arbeitsumgebung schaffen, die sie vor allem in den ersten Jahren ihrer Arbeitsverhältnisse brauchen, wird ebenfalls angestrebt".

3 Vergleiche Hemmingsen und Skov (2003), wesentliche Schlagworte sind: „develop software based on the open source development method" und „use of open standards".

Und weiter hinten:

> „Es wurden Fachleute beauftragt, Lernmanagementsysteme (LMS) zu visitieren und Ansätze für eine Auswahl von Content-Management–Systemen (CMS) auszumachen – hier werden Open-Source-Lösungen bevorzugt".

Auf die beiden letztgenannten Entwicklungen wird noch im letzten Abschnitt einzugehen sein.

4 Open Source im Wirtschaftsgeschehen

Zeitlich synchron zur Diskussion von IT-Welten im Bildungsbereich wurde die klein- und mittelständische Wirtschaft in Österreich auf die Nutzung von Open-Source-Produkten als Entwicklungswerkzeug und Plattform für Anwendersoftware aufmerksam. Die Gruppe junger Wirtschaftstreibender nennt sich „Open Source Experts" und richtete nun bereits zweimal (im Januar 2005 und April 2006) die OSCON-Konferenz[4] aus.

Die JungunternehmerInnen wollen sich die Lissabonziele der Europäischen Union (weltweit dynamischster Wirtschaftsraum, nicht zuletzt durch eine Bündelung der nationalen IT-Strategien) täglich neu vornehmen. In Zeiten des anziehenden Wirtschaftswachstums und von Exporterfolgen der österreichischen Wirtschaft in Osteuropa (dem die Binnennachfrage noch nicht so folgen kann) soll sich auch eine kleine und feine Softwareindustrie herausbilden, die mit Open-Source-Produkten zunehmend konkurrenzfähig wird. Offene Standards, offene Datenformate, offener Zugang zu Netzwerken und Basisdaten können die Grundlage für eine solche Infrastruktur bilden.

Interessante Projekte betrieben in diesem Zusammenhang die öffentlichen Bedarfsträger (öffentliche Versicherungen, staatliche Geld- und Kreditangebote): Die Ausrichtung war es, die bestehenden Systemlandschaften durch eine moderne, offene und – soweit sinnvoll auf Open-Source-Komponenten beruhende – neue, webzentrierte technische Architektur zu ergänzen.

Der Einsatz von Open Source in der Sozialversicherung wurde im Projekt „eSV – Elektronische Sozialversicherung im Internet" erfolgreich umgesetzt. Die *e-card* ist der zentrale Schlüssel zu den Leistungen der österreichischen Sozialversicherung und des Gesundheitswesens. Sie ist bereits für den Einsatz als Bürgerkarte vorbereitet. Das Konzept Bürgerkarte wurde erfolgreich mit Open-Source-Komponenten umgesetzt. Nach Aktivierung der Bürgerkartenfunktion bekommt jeder Kunde einen sicheren Zugang zu den elektronischen Dienstleistungen der Sozialversicherung. Mit der Bürgerkarte kann die Kommunikation mit allen Behörden und Unternehmen sicher, rasch

4 Die Webseite der *Open Source Business Conference* in der Wirtschaftskammer Wien ist zu finden unter http://www.oscon.at.

und bequem abgewickelt werden. Österreich hat in der EU immerhin nun den ersten Platz im E-Administration-Bereich (fälschlich oft als E-Government bezeichnet) zu verteidigen.

Durch die breite Einführung von Open-Source-Software in den Kernbereichen der öffentlichen Verwaltung, wie oben bereits angeführt auch in Teilen der Schulverwaltung, entsteht als Folgewirkung langsam der Bedarf, Know-how zum Betrieb dieser Open-Source-Produkte und Arbeitsumgebungen auch im Schulwesen und vor allem der in Österreich recht unmittelbar damit verbundenen beruflichen Ausbildung zu erwerben. Die Komponente gewinnt gegenüber der preisgünstigen Nutzung freier Software an Schulen nur langsam, aber nachhaltig an Bedeutung. Man darf sich hier allerdings keinen Illusionen hingeben: Genau mit diesem Argument, der unbedingten Praxisnähe, wird auf die Erfahrung mit kommerziellen Produkten bereits im Schulunterricht verwiesen. Wird der Einsatz von Open-Source-Produkten in der durch Klein- und Mittelbetriebe dominierten österreichischen Privatwirtschaft breiter, tun sich auch die Befürworter dieses Einsatzes in Schulwesen leichter. Nicht zuletzt aus diesem Grund darf sich der Bildungsbereich im Rahmen der Zusammenkünfte der Wirtschaftstreibenden immer wieder präsentieren – als eine Art Hoffungsträger für zukünftige Exporterfolge.

Allgemein lässt sich seit 2004 – die OSCON-Konferenzen hatten 2004 einen Vorgänger, den *E-Day* der EDV-Wirtschaft – feststellen: Die IT-Entwicklungsstrategien werden sachlicher, einige sind mit Desktop-Systemen schon auf die Nase gefallen und fahren zwei EDV-Welten, die Art der messianischen Bekehrung für eine Ideologie des Softwareeinsatzes geht zurück. Es bestehen gute Chancen, mit balancierten Konzepten und nüchterner Betrachtungsweise echte Problemlösungen zu entwickeln, die nicht nur wegen des Trademarks *Open Source* gut sind.

5 Eine nachhaltige Linux-Initiative ab 2005

Wirtschaftskontakte, Studien zu „Linux an den Schulen" und ein gutes Maß an nüchternen Fakten – eine Initiative, die sich nur auf eine Insiderperspektive stützt und als Träger einzelne Experten hat, die ihrem Individualismus frönen, und nicht weit kommt. Eine Kooperation mit einer großen Firma war notwendig, trotz oder gerade wegen der vielen Editionen und Möglichkeiten im Open-Source-Bereich. Es traf sich gut, dass *Novell* 2005 *SUSE* gekauft hatte und mit *Novell* bezüglich der Netzwerksoftware traditionell seit längerer Zeit gute Kontakte bestanden.

Also wurde im Oktober 2005, auf Basis der Kremser Studie „Linux an den Schulen", ein Abkommen mit *Novell* geschlossen, das für Schulen folgende Serversoftware zugänglich macht: *Novell Open Enterprise Server* inkl. *SUSE Open Enterprise Server*, *ZENworks Desktop Management*, *ZENworks Linux Management* (verteilte Systeme), *Groupwise* (Collaboration) und *Bordermanager* (Firewall). Diese dritte Open-Source-Initiative fand rasch Anklang – 54 Schulen und 35 000 SchülerInnen

stiegen anfangs ein und stellten ein solides Entwicklungsprojekt dar. Von Vorteil war dabei die begleitende Lehrerinitiative, *n4education*[5] genannt, die sich der „Bedeutung der funktionierenden, leistungsfähigen und sicheren Computernetze und der damit zusammenhängenden umfangreichen Systembetreuungsarbeit" widmet. Die Initiative ist mit Novell-Produkten verknüpft und hat die Betreuung der Netzwerkinfrastruktur an den Schulen zum Ziel. *n4education* organisiert Schulungen und Informationsveranstaltungen zu den oben genannten Produktgruppen. Daher ist es kein Zufall, dass sich die Anzahl der Schulstandorte, die nun mit Stand November 2006 mitmachen, auf über 100 erhöht hat.

Die Grazer und Tiroler Kollegen haben, passend zum *Novell* bzw. *SUSE Open Enterprise Server*, ein Desktop-Produkt, basierend auf *OpenSUSE Linux 10.1*, auf den „Markt" gebracht. Die Initiative nennt sich wenig überraschend „desktop4education" und enthält auch *OpenOffice.org*, *Staroffice 8* und viele nützliche Applikationen.[6] Die CD wurde vom Bildungsministerium bis Ende November 2006 an alle weiterführenden Schulen verteilt. Es ist keine Frage, dass sich 2006 weitere Höchststände der im zweiten Abschnitt dargestellten Zahlenreihen bezüglich einer Verwendung von Linux-Derivaten ergeben werden.

6 Open-Source-Entwicklungen im Lernprozess – da wird es wirklich spannend

Software zur Unterstützung von Lernprozessen hat in den letzten Jahren ein eigenes Profil entwickelt: Community-orientiert, leicht adaptierbar und deutlich entfernt von den Lebenszyklen kommerzieller Software. Letztere braucht ein hohes Maß an Standardisierung, große Stückzahlen oder eine längerfristige Einsatzzeit, um eine maßgeschneiderte Systemlösung kostendeckend zu machen. Schule hingegen ist ein experimenteller Raum mit vielen Adaptierungswünschen, immer mehr propagierten individuellen Lernpfaden und geringer „Kostenwahrheit" – ideal für den Einsatz von freier Software. Im Zentrum des Lernens werden in Zukunft nicht Standardsoftwareprodukte sein, die zufällig die Bürowelt nachbilden, sondern Lernplattformen und Content-Management-Systeme für ganz spezifische Einsatzbereiche.

Österreichs Schulen beschäftigen sich seit ca. fünf Jahren mit der Nutzung von Lernplattformen für den täglichen Unterrichtseinsatz. Zwei Projekte dazu heißen „eLearning im Schulalltag"[7] und „eLearning-Cluster" (Dorninger et al. 2003). Viel wurde untersucht, viel experimentiert und es wurde so manche Entwicklung zur Neugestaltung von Lernplattformen noch vor drei Jahren in Auftrag gegeben. Das Open–Source-LM-System *Moodle*[8], von australischen Pädagogen mitentwickelt, überrollt

5 http://www.n4e.at
6 http://www.d4e.at
7 http://elsa.schule.at
8 http://www.edumoodle.at/moodle/

sie alle – und zwar in Rekordzeit. Während sich gut strukturierte regionale Bildungsserver in einigen Bundesländern noch halten können, erobert *Moodle* alle Schulserver und ermöglicht durch neue Plugins und neue Routinen eine an den Lernfortschritt angepasste Adaptierung der Arbeitsformen in virtuellen Lernräumen. Die Moodle–Community entwickelt, was Lehrende mögen und brauchen können. Damit gibt es im österreichischen Schulwesen auf dem LMS-Sektor keine kommerziellen Produkte mehr (auf Universitätsboden ist dies durch gute Plattformen, in die viel investiert wurde, wie *WebCT*, *Blackboard* u. a. etwas anders). Das Moodle-LM-System wurde nun auch seit April 2006 unter dem Titel „eduMoodle-Server" als zentraler Service für einen „Testbetrieb" für 50 Schulen zur Verfügung gestellt. Mittlerweile sind es 300 unterschiedliche Schuldomänen und man muss sich über die Performance der Host-Serverfarm wirklich Sorgen machen. Die Anzahl der Moodle-Projekte ist kaum überschaubar, in Meetings und Kongressen versucht man, einen ungefähren Überblick zu behalten.

Schüler und Lehrer wurden im Rahmen einer Studie zur Nutzung des *content* an Österreichs Schulen befragt, wie sie ihre Materialien auf welchen Servern finden (Prähauser-Hornung et al. 2006). Obwohl die Bildungsserver und Lernportale gut entwickelt sind und gepflegt werden, war die Antwort bei den Schülerinnen und Schülern eindeutig: *Google* vor *Wikipedia* und erst dann die erste Plattform *schule.at*. Lehrende hingegen vertrauen eher auf die Angebote der eingeführten Bildungsserver.

Damit werden zwei unterschiedliche Zugänge bestätigt, die in Österreich um 2003 diskutiert wurden: Ist es notwendig, betulich „Bildungsserver" und abgezirkelte Webangebote, wenn möglich, mit „nationaler Identität", aufzubauen, um den Bildungsinstitutionen ein System von Wissen und Austauschplattformen für Meinungen zukommen zu lassen? Oder genügt es einfach – so unter der Devise „es ist alles im Web, man muss es nur finden" – die vorhandenen Angebote im Web gut zu vernetzen und ein nationales *repository* für Anwender aufzubauen, um die verstreuten Informationen zielgerichtet finden zu können? Damals wurde für Österreichs Schulen ein Mittelweg beschritten und der so genannte „Content-Erlass" (Bundesministerium für Bildung, Wissenschaft und Kultur 2003) als Rahmenrichtlinie für Webangebote (Entwicklung, Providing, Distribution von elektronischen Lehr- und Lernmaterialien) herausgegeben. Die Devise war: Bildungsserver werden gepflegt und können irgendwo errichtet werden; man konzentriert sich auf ein gutes *repository* für die Anwender, eine aktuelle Metadatenbeschreibung und wirkt bei europäischen Projekten mit (*Celebrate*, *Calibrate*, *MELT*). Der Weg geht nun zur Nutzung dieser Angebote und von Wissens-Serviceleistungen der offenen Web-Communitys (wie *Wikipedia*).

Durch die Webpräsenz von brauchbaren Inhalten und die zahlreichen Serviceleistungen im Internet hat sich der Arbeitsstil der jungen ProjektarbeiterInnen vor und nach (sekundärem oder tertiärem) Ausbildungsabschluss dramatisch geändert: Wer nicht im Web suchen gelernt hat und Angebote von Kommunikationsplattformen u. a. zu nutzen vermag, hat wenige Chancen am Arbeitsmarkt der jungen Arbeit-

nehmer. Schule und Universität bieten vielfache Möglichkeiten, diese „alternativen" Projektarbeitsformen kennen zu lernen und zu pflegen. Bereits recht junge Leute entwickeln in der Nutzung dieser Instrumente eine oft bemerkenswerte Virtuosität: Im österreichischen berufsbildenden Schulwesen trifft man immer wieder auf 17-Jährige, die als Projektmanager neben der Ausbildung erste größere Entwicklungsprojekte (z. B. im Software-Engineering) mit weltweit verstreuten (oft anonymen) Communitys bearbeiten und oft nebenher auch erstes Geld damit verdienen. Auch in Zeiten prekärer Anstellungsverhältnisse oder nur sehr entfernt winkender fixer Arbeitsverhältnisse für unsere Absolventinnen und Absolventen muss klar sein, dass diese weborientierten Arbeitsformen die eigentliche neue Berufsausbildung wesentlich mitbestimmen. Die Wurzel dieser Projektarbeitsformen sind die Foren, Wiki-Listen und Blogs der Open-Source-Communitys, wie sie Mitte der 90er Jahre begonnen haben. Sie wurden nur unwesentlich adaptiert und bestimmen das Zusammenarbeiten im Beschäftigungsbereich. So nahe war die Lernarbeit in Bildungsinstitutionen noch nie an der Arbeitsorganisation im Beschäftigungssystem dran.

Ein wesentlicher Begriff bei der Neukonstruktion des Lernens heißt „Social Software" – der Entwurfsprozess über selbst gesteuerte Wiki-Listen, die Verwendung von Fach-Weblogs und die Darstellung von Ergebnissen in E-Portfolios. Diese und weitere Instrumente werden den „neuen Lernprozess" entscheidend prägen und für jede Art von Bildung und Ausbildung wichtig werden. E-Portfolios als persönliche, überall verfügbare digitale Leistungsmappen entsprechen den Anforderungen der europäischen Bildungspolitik im Hinblick auf das „lebensbegleitende Lernen".[9] Die grundlegenden Ideen dazu und die Realisierung von ersten technischen Lösungen sind wohl in den Open-Source-Communitys zu finden. So wie überhaupt „Social Software" eine Domäne der offenen Standards, offenen Plattformen und einer GPL-Lizenz-Nutzung ist; die kommerziellen Anwendungsmöglichkeiten halten sich ja in Grenzen.

Die Transformation von „elektronischer Bildung in Dosen" (E-learning-Angebote und Kurse) zur Web-Gemeinschaft mit vielen Ideen und frei nutzbaren Umgebungen und Instrumenten wird immer deutlicher. Faktum ist, dass in vielen Schulsystemen und Bildungsverwaltungen in Europa vor gut 20 Jahren die Entscheidung getroffen wurde, mit kommerziellen Betriebssystemen und den gerade am Markt befindlichen Standardsoftwareprodukten „Elektronische Datenverarbeitung" als Schulfach, Unterrichtsprinzip oder Neigungsgruppe einzuführen. In Österreich war dies 1985. Seitdem haben Internet- und World-Wide-Web-Zugänge die Arbeitsformen modifiziert. Trotzdem steht eine Umstellung im oben beschriebenen Sinn im Bildungsbereich bis zur Sekundarschule erst bevor. Zu groß ist der Widerstand, vor allem auch aus der Lehrerschaft, die liebgewordenen Pfade der klassischen Programmierung oder Bürosoftware-Nutzung zu verlassen. Es wird also eine gleitende Umstellung geben müssen, sowohl den Softwareeinsatz als auch die Arbeitsumgebungen betreffend. Dies

9 Siehe auch „EU-Aktionsplan i2010" oder „EU-Aktionsplan für die Entwicklung der allgemeinen und berufsbildenden Bildung bis 2010".

bedeutet auch eine sanfte Migration von kommerzieller Standardsoftware zu freier, soziale Prozesse fördernder Software. Komponenten dieser sanften Migration sind grundsätzliche Überlegungen zum Status der „Elektronischen Datenverarbeitung" in einem Unterrichtscurriculum, die Bewusstseinsbildung unter den Schulpartnern, einer Strategie der Plattformunabhängigkeit bei neuen Beschaffungen und Entwicklungen, von Parallelinstallationen, wie im Abschnitt 3 beschrieben, und entsprechenden Schulungen und Evaluierungen. In gut zehn Jahren wird der Nutzung von freien Softwareumgebungen im Bildungsbereich die Zukunft gehören.

Literatur

Bundesministerium für Bildung, Wissenschaft und Kultur (2003), 'Elektronischer Content für Österreichs Schulen –Entwicklung, Verbreitung und Pflege', Content-Erlass Zl. 16.700/164-II/8/2003. http://www.bmbwk.gv.at/medienpool/10108/eContent_Schulen.pdf [16. Jan 2007].

Dorninger, C., Horschinegg, J., Cortolezis-Schlager, K., Schubert, K., Schnalzer, C. und Schrack, C. (2003), 'eLearningCluster: Allgemeine Informationen zum Projekt eLearningCluster', Bundesministerium für Bildung, Wissenschaft und Kultur. http://www.e-teaching-austria.at/eLC/index.html [16. Jan 2007].

Hemmingsen, M. und Skov, A. (2003), 'The Danish Software Strategy', Danish Ministry of Science, Technology and Innovation. http://www.oio.dk/files/Softwarestrategi_-_Engelsk.pdf [16. Jan 2007].

IT-Lenkungsgruppe im Bundesministerium für Bildung, Wissenschaft und Kultur (2005), 'Online-Befragungen von Schulen zur Hard- und Softwareausstattung', Nicht veröffentlicht. http://www.efit.at/infrastruktur/projekt.asp?Projektname=IT-Monitoring [19. Jan 2007].

Menzel, T. und Psihoda, R. (2004), 'eFit Austria – die neue Qualität der Bildung', Bundesministerium für Bildung, Wissenschaft und Kultur. Broschüre, Auszüge unter http://www.efit.at [16. Jan 2007].

Prähauser-Hornung, V., Mayringer, H. und Hilzensauer, W. (2006), eContent und SbX – Bekanntheit und Nutzung webbasierter Lehr- und Lernmaterialien an Schulen, Studie, Salzburg Research Forschungsgesellschaft, Salzburg. http://www.bildung.at/content/bmbwk_studie_econtent.pdf [19. Jan 2007].

Raymond, E. S. (2001), *The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary*, 2. Aufl., O'Reilly Media, Cambridge, MA, USA.

Schwed, G. (2005), 'Linux an österreichischen Schulen', e-LISA academy Newsletter. http://www.e-lisa-academy.at/?design=elisawp&url=community&cid=7849&modul=10&folder=60954 [19. Jan 2007].

Einsatz von Open-Source-Software in der Entwicklungszusammenarbeit – das Beispiel Afghanistan

NAZIR PEROZ, JAN SUHR UND DANIEL TIPPMANN

(CC-Lizenz siehe Seite 563)

Häufig werden im Zusammenhang mit Open-Source-Software (OSS) und Entwicklungsländern die Vorteile von OSS gegenüber proprietärer Software hervorgehoben. Die Erfahrungen der drei Autoren mit Open-Source-Software in zwei universitären Aufbauprojekten in Afghanistan sollen diese Diskussion bereichern. Open-Source-Software war vor den Projekten in Afghanistan quasi unbekannt. Daher mussten Administratoren erst geschult werden. Andererseits war die Administration von Linux-Systemen relativ einfach und deren Sicherheit deutlich im Vorteil gegenüber proprietären Windows–Systemen. Die Vorteile von Open-Source-Software in einer Thin-Client-Architektur[1] können wir somit bestätigen. Auch war die Förderung des Verständnisses von den zugrunde liegenden Technologien ein willkommenes Ergebnis. Abschließend fällt unsere Bewertung nicht uneingeschränkt aber deutlich positiv für den Open-Source-Ansatz aus.

Schlüsselwörter: Entwicklungsland · IT-Ausbildung · Softwarelokalisierung

1 IT in Afghanistan

Die besonderen Eigenschaften von Open-Source-Software und deren Eignung für den Einsatz in Entwicklungsländern wurden bereits vielerorts diskutiert. Die theoretische Argumentation für die Vorteile von Open-Source-Software in diesem Zusammenhang

1 Hierbei dienen die PCs, an denen die Benutzer arbeiten, nur als schlanke Endgeräte zur Ein- und Ausgabe (Maus- und Tastatureingaben bzw. Bilddarstellung), während ein zentraler Server, der über das Netzwerk mit den Clients verbunden ist, die eigentliche Arbeit verrichtet (Ausführen der Programme, Verteilung der Ressourcen). Auf diese Weise arbeiten die Benutzer quasi „auf dem Server" und bekommen den Desktop auf ihren PC übertragen (eine Art *remote desktop*).

mag oft überzeugend sein, doch fehlt es bislang an praktischen Bestätigungen für diese Annahmen. Dieser Artikel soll hierzu einen Beitrag leisten, indem unsere Erfahrungen bei der Verwendung von Open-Source-Software in Afghanistan dargestellt werden.

Afghanistan ist in besonderer Weise vom Rückstand in Bezug auf Informationstechnologie (IT) betroffen. Während der Kriegs- und Bürgerkriegszeiten im Land, die mehr als 20 Jahre andauerten, hat die Weltgemeinschaft durch die explosionsartige Verbreitung der IT einen Strukturwandel erfahren, der in dramatischer Weise alle gesellschaftlichen Bereiche erfasste, an Afghanistan aber vorbeigegangen ist.

Zum Zeitpunkt des Petersberger Beschlusses bei Bonn[1], im Dezember 2001, hatte Afghanistan keinen Anschluss an die Informationsgesellschaft. Damals hatten zwischen 22 und 60 Prozent der Bevölkerung Europas und der USA Zugang zum Internet.[2] Seit den vergangenen fünf Jahren entwickelt sich die IT in Afghanistan explosionsartig und die Nachfrage nach dieser Schlüsseltechnologie, die während mehr als 20 Jahren Krieg und Bürgerkrieg der Bevölkerung Afghanistans verwehrt war, steigt rapide im Zuge des Wiederaufbaus.

2 Unsere Projekte in Afghanistan

Das Zentrum für internationale interkulturelle Kommunikation (ZiiK) der Technischen Universität Berlin (TU Berlin) ist seit dem Jahr 2002 mit einer Reihe von IT-Projekten in und für Afghanistan tätig. Dazu gehören die Errichtung von Rechnerzentren, Aufbau von Informatik-Fakultäten[3], Durchführung von Konferenzen, Workshops und Austauschprogrammen, Beratertätigkeiten etc.[4] Als wichtigste Projekte, auch und vor allem im Zusammenhang mit Open Source, sollen die beiden folgenden kurz vorgestellt werden.

2.1 IT-Center Kabul (ITCK)

Unmittelbar nach dem Sturz der Taliban-Regierung im Jahr 2001 begannen unsere Verhandlungen mit offiziellen Stellen, Universitäten und möglichen Geldgebern für IT-Aufbauprojekte in Afghanistan. Der „Stabilitätspakt Afghanistan" sah mit Unterstützung der Bundesregierung eine langfristige Zusammenarbeit vor und im Au-

1 „Vom 27. November bis zum 5. Dezember 2001 fanden auf dem Petersberg bei Bonn die von den Vereinten Nationen geleiteten Gespräche zu Afghanistan statt. Die Konferenz endete mit einer 'Vereinbarung über provisorische Regelungen in Afghanistan bis zum Wiederaufbau dauerhafter Regierungsinstitutionen' (Bonner Vereinbarung). Die Vereinbarung sieht vor, dass eine vorläufige Regierung am 22. Dezember 2001 offiziell die Regierungsgewalt in Afghanistan übernimmt." (Deutscher Bundestag 2001)

2 Vgl. die Statistiken zur weltweiten Internetnutzung von ITC unter http://www.itu.int/ITU-D/icteye/Indicators/Indicators.aspx# v. a. zum *digital divide*: http://www.itu.int/ITU-D/ict/statistics/ict/graphs/internet.jpg.

3 Unter anderem an den Universitäten in Kabul und Herat. Projekte in weiteren Städten sind geplant.

4 Siehe http://projekte.ziik.org/afghanistan für einen Überblick über unsere Projekte.

gust 2002 wurde ein fünfköpfiges Team der TU Berlin unter der Leitung von Herrn Dr. Peroz zur Evaluation der Lage nach Kabul geschickt. Die Universität Kabul verfügte zu dieser Zeit über eine sehr geringe, schlechte Computerausstattung, jedoch bestand großes Interesse, diese Technologie einzusetzen. Aufbauend auf den Ergebnissen der Gespräche in Kabul und der Erfahrung der TU Berlin im Arbeitsgebiet „Informatik und Entwicklungsländer" wurde ein Gesamtkonzept zur IT-Versorgung für die Universitäten Afghanistans am Beispiel der Universität Kabul erarbeitet und an den Deutschen Akademischen Austauschdienst (DAAD) weitergereicht. Das Konzept basierte auf einem dreistufigen Plan:

- Aufbau des Rechnerzentrums (RZ)
- Aufbau von Informatik-Fakultäten
- Aufbau weiterer Fakultätsnetzwerke

Das Rechnerzentrum wurde als Herzstück der IT-Versorgung geplant, von dem IT–Dienstleistungen zentral für die Universität erbracht werden. Als „IT-Center Kabul" (ITCK) soll es ein Kompetenzzentrum und damit Vorbild für den Aufbau aller weiteren Universitäten im Land sein und eine Vorreiterrolle im Prozess der landesweiten Versorgung mit IT einnehmen. Im nächsten Schritt werden die dringend benötigten personellen Ressourcen durch den Aufbau von Informatik-Fakultäten geschaffen. Daran anknüpfend können schließlich fachspezifische Belange von allen Fakultäten der Universitäten durch Errichtung von PC-Pools und Netzwerkstrukturen bedient werden. Im Folgenden wird auf unsere Erfahrungen beim Aufbau des ITCK eingegangen.

Für das ITCK wurden von der Universitätsleitung mehrere Räume in einem Gebäudekomplex zur Verfügung gestellt. Bei der Projektarbeit mussten, aufgrund der örtlichen Gegebenheiten, Faktoren wie eine zuverlässige Stromversorgung, staub- und feuchtigkeitsfreie Räume, Überwindung der Sprachbarriere, nötiges Fachpersonal etc. bedacht werden. Ein Internetanschluss mit einer für Afghanistan relativ hohen Bandbreite (derzeit ca. 1 MBit/s) wurde von der *NATO* über eine Satellitenverbindung realisiert. Das Konzept sah eine Ausstattung von 70 Arbeitsplatz-PCs, zwei Servern, vier Sprachlaboren, einem entsprechend ausgestatteten Serverraum, Seminarräumen, Frauenraum, Konferenzraum etc. vor. Damit können neben der Bereitstellung von Internet-Arbeitsplätzen auch Schulungen und Weiterbildungen im Bereich IT durchgeführt werden. Außerdem wurde mit dem Aufbau einer Informatik-Bibliothek begonnen. Die Hardware wurde in Deutschland beschafft, die Software installiert und dann nach Kabul transportiert.

Als Betriebssystem für die Arbeitsplatz-PCs wurde Windows XP, für die Server *Debian GNU/Linux* ausgewählt. Dies geschah aufgrund einer Reihe von Überlegungen: Generell startete das Projekt mit dem Tenor, nicht nur eine Versorgung mit Technologie und einen Wissenstransfer zu leisten, sondern auch den Open-Source–Gedanken nach Afghanistan zu transportieren. Neben der politischen Signalwirkung

bietet der Einsatz von Open-Source-Software gerade in strukturschwachen Regionen eine Reihe von Vorteilen, die mittlerweile einschlägig bekannt sind und, soweit es unsere Projekte betrifft, in den folgenden Kapiteln im Einzelnen näher ausgeführt werden.

Afghanistan hatte zu diesem Zeitpunkt jedoch wenig bis keine IT-Fachexpertise, die für einen erfolgreichen Einsatz von Open Source notwendig ist. Wo sie vorhanden war, bestand sie im Wesentlichen aus Grundkenntnissen im Umgang mit *MS Windows* und *MS Office*. Um diese Fachkenntnisse zu unterstützen und auszuweiten, sollte den Benutzern ITCK zunächst eine Arbeitsumgebung zur Verfügung gestellt werden, die ihnen zumindest grundlegend vertraut ist, weswegen die Wahl für die Client-PCs auf Windows fiel. Ein weiterer Punkt war die Verfügbarkeit verschiedener lokalisierter Software, die nur für Windows zur Verfügung steht. Die Hauptsprachen in Afghanistan sind *Dari* und *Pashtu* und die wenigen Lehr- und Lernprogramme in diesen Sprachen sowie die einschlägigen Wörterbücher und Lexika waren bis dato nicht für andere Plattformen erhältlich.

Im Bereich der Server hingegen, wo auch eine intensivere Schulung und längerfristige Betreuung der Administratoren geplant war, wurde durchgängig Open-Source-Software eingesetzt. Vor dem Hintergrund der technischen Vorteile, der Zuverlässigkeit und der Zukunftssicherheit und damit der Nachhaltigkeit von Linux[5] bei gleichzeitiger Lizenzkostenminimierung boten sich kaum Alternativen hierzu.

Damit entstand darüber hinaus in Bezug auf die technische Plattform eine relativ heterogene IT-Landschaft, die eine friedliche Koexistenz beider Ansätze – proprietär und Open Source – signalisierte. Dadurch, dass weder die eine, noch die andere Seite ausgegrenzt wurde, konnten wir den Studierenden und anderen Nutzern des ITCK die Chance geben, mit beiden Welten vertraut zu werden. Ob dieser Gesamtansatz seine beabsichtigte Wirkung zeigt, wird sich erst im Lauf der Zeit zeigen.

2.2 PC-Pool der Informatik-Fakultät an der Universität Herat

Als zweites großes Projekt startete 2004 der Aufbau einer Informatik-Fakultät an der Universität Herat. Seit jeher gilt Herat als wichtiger Bildungsstandort in Afghanistan und die dortige Universität wird von mehreren Hilfsorganisationen unterstützt. Im Wintersemester 2004/2005 begann ein Team der TU Berlin mit der Realisierung des Projekts. Bereits im zweiten Semester (Sommer 2005) wurden über 200 Studierende an der Fakultät nach einem an der TU Berlin entwickelten Informatik-Curriculum unterrichtet.[6]

Zentraler Bestandteil des Projekts war die Errichtung eines PC-Pools für die Fakultät in Herat, in dem die Studierenden ihr erlerntes Wissen praktisch in die Tat

5 In der Vergangenheit gab es eine lebhafte Diskussion um die Namensgebung dieses Systems. Im Sinne der besseren Lesbarkeit verwenden wir im Text nur die Bezeichnung Linux anstelle von *GNU/Linux*; gemeint ist jeweils das Gesamtsystem bestehend aus Linux-Kernel und zugehöriger Software.

6 Für weitere Informationen zu diesem Projekt siehe Mahr et al. (2006).

umsetzen können. Aus einer Spende an die Universität war bereits ein Bestand von ca. 15 verhältnismäßig aktuellen PCs vorhanden. Diese Ausstattung sollte zu einem vollwertigen PC-Pool ausgebaut werden. Hierzu wurde ein Thin-Client-Konzept entwickelt und umgesetzt.

Als Plattform für dieses System kam fast ausschließlich Open-Source-Software auf Linux-Basis zum Einsatz. Lediglich ein zusätzlicher Terminal-Server unter Windows, auf dem nur für diese Plattform verfügbare Software genutzt werden kann, wurde transparent integriert. Auch für dieses Projekt wurde das Equipment in Berlin beschafft und konfiguriert, anschließend nach Afghanistan transportiert und dort von einem Team der TU Berlin aufgebaut und in Betrieb genommen. Ein Internetanschluss wurde vor Ort über eine Punkt-zu-Punkt-WLAN-Verbindung zu einem lokalen Provider realisiert. Diese Verbindung war allerdings relativ langsam und unzuverlässig.

3 Wirtschaftlichkeit

Im Folgenden soll nun auf einzelne Aspekte von Open Source im Vergleich zu proprietärer Software unter Berücksichtigung der Erfahrungen aus diesen beiden Projekten eingegangen werden.

3.1 Lizenzkosten und gelebtes Urheberrecht

Häufig wird von Open-Source-Befürwortern das Argument genannt, dass für proprietäre Software einmalige bzw. regelmäßige Lizenzgebühren zu entrichten seien, die Entwicklungsländer aufgrund ihrer geringen Kaufkraft nicht aufbringen könnten. Open-Source-Software sei hingegen meistens kostenlos. In Afghanistan wird nach unserer Beobachtung beinahe jede verwendete proprietäre Software nicht über den vom Softwarehersteller gewünschten Vertriebsweg erhalten, sofern sie nicht von Dritten finanziert, als Spende zur Verfügung gestellt oder zu einem PC mitgeliefert wurde. Es existiert kein Urheberrecht, welches das Kopieren proprietärer Software verbietet. Daher ist es nicht verwunderlich, dass Softwaregeschäfte fast ausschließlich Raubkopien verkaufen, die üblicherweise einige Cent bis wenige Dollar kosten. Diese werden auch in Behörden und anderen staatlichen Organisationen verwendet.

Die Aufbauorganisation *Peace Corps* kam in einer Untersuchung zu dem Ergebnis, dass die Installation proprietärer Software in sich entwickelnden Ländern beiträgt, vom Hersteller nicht autorisierte Kopien zu verwenden (Krotoski 2005). Diese Erkenntnis können wir aufgrund unserer Erfahrungen in den hier beschriebenen Projekten bestätigen. In Afghanistan sind unautorisierte Kopien von Software nicht illegal und daher besteht bei deren Verwendung selbstverständlich auch kein Unrechtsbewusstsein beim Benutzer. Und selbst wenn dies doch der Fall sein sollte, existiert keine realistische Möglichkeit, eine vom Hersteller autorisierte Kopie zu einem bezahlbaren Preis zu erwerben. In unseren Projekten bedurfte es daher einiger Überzeugungsarbeit, die Administratoren dazu zu bewegen, die von uns bereitgestellten legalen und definitiv

virenfreien Versionen proprietärer Software anstatt vom Hersteller nicht autorisierter Kopien zu verwenden. Umgekehrt führte die Installation von Linux und anderer Open-Source-Software zu derer Verbreitung. Viele Studenten tauschten Linux-CDs untereinander und installierten dieses Betriebssystem auf ihren eigenen Computern. In einigen Software-Geschäften in Herat und Kabul sind mittlerweile Linux-CDs erhältlich.

Sowohl für das ITCK als auch für die Informatik-Fakultät in Herat wurden von *Microsoft* kostenlose, aber zeitlich limitierte Lizenzen für Windows und Office bereitgestellt. Hätte die Software regulär bezahlt werden müssen, so hätte dies einen großen Teil des verfügbaren Budgets gekostet.[7] In diesem Falle wäre das Konzept für das ITCK vermutlich auch zu Gunsten von Open-Source-Software auf den Workstations ausgefallen.

An der Informatik-Fakultät in Herat kam *SUSE Linux* als Server und *Astaro Security Linux* als Firewall-System zum Einsatz. Trotz kostenloser Alternativen fiel die Wahl auf kostenpflichtige Linux-Distributionen, da diese insbesondere für unerfahrene Administratoren einfacher zu bedienen sind. Im Falle der Firma *Astaro*, die uns Sonderkonditionen für ihre Linux-basierte Firewall anbot, genügte ein Versprechen unsererseits, den Produktnamen in öffentlichkeitswirksamen Dokumentationen und Berichten des Projekts zu nennen, um das Produkt kostenfrei verwenden zu können. Zusammenfassend lässt sich sagen, dass nach unseren Erfahrungen den Lizenzkosten für proprietäre Software in Afghanistan momentan eine geringe Bedeutung zukommt und eine Veränderung dieses Umstandes nicht abzusehen ist.

3.2 Administrations- und Benutzerfreundlichkeit

Der Gesamtaufwand für die Administration und den Support von IT-Systemen wurde von diversen Studien (Robert Francis Group 2002; Höhn und Herr 2004; Becta 2005; Wheeler 2005) bei der Verwendung von Linux geringer als bei Windows eingeschätzt, wobei teilweise auch gegenteilige Ergebnisse ermittelt wurden (Bozman et al. 2002; Wheeler 2005). Bei unseren Projekten in Afghanistan verfügten nicht alle, aber die Mehrheit der Benutzer und Administratoren über Grundkenntnisse in der Benutzung von Windows und anderen Microsoft-Produkten. Häufig wurde die Benutzung dieser Systeme in einem der Nachbarländer erlernt, wo viele der Betroffenen während der Kriegsjahre als Flüchtlinge lebten und teilweise zur Schule gingen. Folglich mussten die Benutzer und die Administratoren an der für sie neuen Open-Source-Software (um-)geschult werden (vgl. Bozman et al. 2002).

In unserem Projekt in Herat, wo Linux-Desktops zum Einsatz kamen, fiel die Benutzung den normalen Anwendern hingegen überraschend leicht, da sie ihre Windows-Kenntnisse ohne große Anstrengung adaptieren konnten. Dies betraf sowohl

7 Offen bleibt, ob wir als eine deutsche Organisation unautorisierte Kopien proprietärer Software in Afghanistan verwenden dürften.

die Benutzeroberfläche des Betriebssystems als auch Webbrowser, Office-Anwendungen und Programmierwerkzeuge. Benutzer ohne Computerkenntnisse hatten kaum Probleme, sich autodidaktisch oder mit Unterstützung ihrer Kommilitonen in die Systeme einzuarbeiten.

Vor der Installation des PC-Pools nach dem oben beschriebenen Thin-Client-Konzept gab es an der Informatik-Fakultät in Herat einen Altbestand an Windows-PCs. Um diese zu nutzen, war es für die Arbeit im Pool vernünftig, auf allen PCs die gleiche Software zur Verfügung zu haben. Die Zusammensetzung der Software auf den Rechnern und auch die Konfiguration änderte sich im Lauf der Zeit jedoch regelmäßig. Eine manuelle Anpassung war zeitaufwändig und fehleranfällig, sodass wir zu einer automatischen Softwareverteilung übergingen. Das bisher verwendete Windows stellte sich hierbei als nicht einfach zu handhaben heraus. Obwohl es prinzipiell möglich war, all unsere Anforderungen mit Windows umzusetzen, ergaben sich diverse Probleme bei der automatisierten Softwareverteilung, der Konfiguration von Software und der Vergabe von Benutzerrechten. Nach einigen Wochen entschieden wir uns, die Windows-Installationen durch Linux zu ersetzen. Schnell stellte sich eine auf unsere Verwendungszwecke zugeschnittene Distribution als einfacher zu administrieren heraus als Windows (Mahr et al. 2006).

Nach Ghosh (2003) machen Lizenzkosten für Software in Industriestaaten 5–10 Prozent an den Gesamtkosten der IT aus. Es könnte also angenommen werden, dass in sich entwickelnden Ländern, in denen die Personalkosten gering sind, sich die Verhältnisse umkehren, so dass die Lizenzkosten proprietärer Software für den größten Teil der Gesamtkosten der IT verantwortlich wären (Ghosh 2003). Daher mag es für sich entwickelnde Länder umso vorteilhafter sein, kostenlose Open-Source-Software zu verwenden (Ghosh 2004, S. 5, S. 13; Rajani et al. 2003). In Afghanistan machten wir die Erfahrung, dass Linux und selbst gängige Open-Source-Software für Windows weitestgehend unbekannt waren. Es gab dort keine Administratoren mit Erfahrung im Bereich Open Source. Daher musste unsere Ausbildung diesen Bereich besonders berücksichtigen. Der dadurch entstandene Aufwand war erheblich und machte einen großen Teil der Projektkosten aus (vgl. Bozman et al. 2002). Das Ziel unserer Projekte war es, eine effektive und zeitgemäße Lehre zur Verfügung zu stellen und so einen nachhaltigen Beitrag zur Entwicklung des Landes zu leisten. Durch die Einbeziehung von Open Source in die Ausbildung sollten Administratoren künftig in die Lage versetzt werden, Open-Source-Software selbständig zu verwenden. Daher ist mittelfristig davon auszugehen, dass Administratoren für Linux auf dem afghanischen Arbeitsmarkt verfügbar sein werden.

Als Nachteil von Linux bzw. Open-Source-Software allgemein wird oftmals die Komplexität angeführt. Somit sei Open-Source-Software für Anfänger wegen des hohen Einarbeitungsaufwands ungeeignet. In der Tat war die Ausbildung nicht immer unproblematisch. Es ließ sich jedoch, vor allem im Herat-Projekt, durchaus ein positiver Nebeneffekt bei der Ausbildung der Administratoren an Linux-Systemen

feststellen. Durch das intensive, bald spielerische Beschäftigen mit Linux und dessen Architektur und Struktur erschlossen sich den Studierenden dabei aber oftmals „nebenbei“ komplexe Zusammenhänge und Hintergründe aus der Netzwerk- und Computertechnik. Diese Kenntnisse konnten sie dann mitunter auch auf andere Bereiche übertragen.

Gerade für die nicht zum Administratoren-Team gehörenden Anwender musste der Einsatz von Linux-Desktops gerechtfertigt werden, da dies eine Veränderung ihrer bisher gewohnten Windows-basierten Arbeitsumgebung bedeutete. Aber auch hier zeigte sich tatsächlich nach kurzer Zeit schon eine recht ungezwungene Nutzung und nach wenigen Wochen war der Umgang mit der neuen Technologie an der Fakultät selbstverständlich geworden. Sehr bald verlangten die Studierenden von uns sogar Linux-CDs und installierten sich das System auf ihren PCs zu Hause, was wir durchaus als Erfolg für das Gesamtkonzept werteten. Mangelnde Benutzerfreundlichkeit kann aufgrund unserer Erfahrungen nicht als Hindernis für die Adaption von Open–Source-Software gesehen werden. Problematisch ist allerdings fehlendes Wissen über die Administration von Open-Source-Software, insbesondere von Linux.

3.3 Hardware-Anforderungen und -Kompatibilität

Bei dem Aufbau von PC-Pools sind die nötigen Hardware-Kosten in die Betrachtung mit einzubeziehen. Wie bereits erläutert, erlaubt es Linux, hochwertige Thin-Client-Lösungen mit langsamen, veralteten Computern und den entsprechenden Servern zu implementieren. Die Vorteile eines solchen Konzepts sind zum einen gute Skalierbarkeit sowie die Möglichkeit, auch ältere PCs mit schwacher Leistung ohne Performanceverlust einzubinden. Zum anderen verringert dieses Konzept den Administrationsaufwand, da lediglich ein Server gewartet und gepflegt werden muss. Gleichzeitig erhöht sich die Sicherheit, da die PCs keine direkte Internetverbindung benötigen, wodurch sich die Angriffsfläche und die nötigen Sicherheitsvorkehrungen wie Firewall, Viren- und Spyware-Schutz reduzieren.

In all unseren Projekten standen letztendlich moderne PCs bzw. Server zur Verfügung, so dass wir nicht mehr auf die Verwendung alter Hardware angewiesen waren, die eine optimale Ausnutzung der Thin-Client-Architektur erlaubt hätte. Lediglich zu Beginn des Herat-Projektes existierten ein Dutzend alter, ressourcenschwacher Pentium-I-PCs. Einige dieser Geräte eigneten sich gut als kleine Server. Der Versuch, die übrigen dieser PCs als Thin-Clients zu verwenden scheiterte allerdings an mangelnder Hardwareunterstützung der Grafikkarte. Der theoretisch bei der Verwendung von Linux mögliche Vorteil der Kostenersparnis durch die Verwendung ressourcenschwacher Hardware wurde in unseren Projekten nicht realisiert.

4 Sicherheit

Sicherheit bedeutet in der IT generell das Gewährleisten von Vertraulichkeit, Integrität sowie die Verfügbarkeit der Daten wie der Systeme und das Durchsetzen von Zugriffsbeschränkungen und Rechten auf diese. Die Sicherheit von Softwaresystemen hat in den letzten Jahren stark an Bedeutung zugenommen. Insbesondere aus dem Internet werden daran angebundene Computer durch eine große Anzahl von Viren, Würmern oder direkt durch menschliche Angriffe bedroht (UNCTAD secretariat 2003, S. 19; Weber 2003, S. 18). IT-Sicherheit spielte bei der Konzeption des Rechnerzentrums und des PC-Pools eine entscheidende Rolle, obwohl die Anforderungen weitaus geringer waren als für IT-Systeme im Produktiveinsatz. Da es noch keine wichtigen Daten zu schützen gab und der Verfügbarkeit noch keine allzu große Bedeutung zufiel, beschränkten sich die Sicherheitsmaßnahmen vor allem auf den Schutz vor Viren, Würmern und anderer Schadsoftware. Obwohl es sich prinzipiell um eine einfache Aufgabe handelt, gab es dabei zahlreiche Probleme. Um permanente Sicherheit gewährleisten zu können, muss entsprechende Antivirensoftware auf allen Windows-PCs installiert und regelmäßig aktualisiert werden. Zwar gab es zu diesem Zeitpunkt gute und teilweise kostenlose Schutzprogramme, doch ermöglichen die kostenlosen Programme i. d. R. keine zentral verteilte Aktualisierung aller PCs über das lokale Netzwerk. Stattdessen sind die integrierten Aktualisierungsfunktionen darauf ausgelegt, die Aktualisierungen für jeden PC aus dem Internet herunterzuladen. Aufgrund mangelnder Internetbandbreite ist dies in Afghanistan i. d. R. nicht praktikabel. Auf personeller Ebene zeigte sich, dass weder Benutzer noch Administratoren ein Bewusstsein für IT-Sicherheit hatten. Daher mussten die Administratoren und auch die Benutzer für die Thematik sensibilisiert werden. Dies beinhaltete u. a. folgende Aspekte:

- Verwendung von geheimen Passwörtern und getrennten Benutzerzugängen
- Vergabe von eingeschränkten Rechten für Benutzer
- regelmäßige Aktualisierung von Antivirensoftware
- regelmäßige Durchführung von Backups
- Art des Umgangs mit persönlichen Informationen bei Mehrbenutzer-Zugängen
- Einsatz von Firewalls
- Umgang mit Malware, Spam, Phishing

Da den Administratoren häufig die nötige Erfahrung und Sorgfalt fehlte und diese oft zeitlich ausgelastet waren, konnte eine ausreichende Wartung nicht immer sichergestellt werden. Linux stellte sich in dieser Hinsicht als robuster heraus, da es dort kaum Viren und weniger Sicherheitslücken gibt. Andererseits führte die Unerfahrenheit der Administratoren im Umgang mit Linux zu sicherheitskritischen Fehlbedienungen, die

bei Windows vermutlich nicht aufgetreten wären. Als weiterer Vorteil stellte sich heraus, dass es im Vergleich zu Windows nicht zwingend nötig ist, für Linux-Systeme eine clientseitige Firewall zu verwenden. Da Firewalls aufwendig zu administrieren sind und häufig Probleme verursachen, erwies sich dies als große Erleichterung. Open-Source–Software weist im Vergleich zu proprietärer Software wesentliche Sicherheitsvorteile auf. Ein entscheidender Faktor ist dabei die fehlende Bedrohung durch Viren für Linux.

5 Nachhaltigkeit

Um den Wartungsaufwand gering zu halten und die Langlebigkeit und Zukunftssicherheit der Lösung zu gewährleisten, sollten verschiedene Aspekte der Nachhaltigkeit wie Interoperabilität und Softwarelokalisierung berücksichtigt werden. Damit wird sich nun im Folgenden auseinandergesetzt.

5.1 Lokalisierung

Global agierende Hersteller von proprietärer Software haben mitunter wenige Anreize, ihre Software auf lokale Bedürfnisse kleiner Regionen u. a. durch Übersetzung in deren Verkehrssprache anzupassen. Bei Open-Source-Software haben die Benutzer die Möglichkeit, die Anpassungen selber vorzunehmen, da technische Hürden niedrig und das benötigte Wissen relativ klein sind (Drakos et al. 2003; UNCTAD secretariat 2003, S. 18). Mit Rosetta existiert sogar eine Webseite, mit der gängige Open-Source–Software übersetzt werden kann, ohne dabei mit technischen Details in Berührung kommen zu müssen.[8]

Das *United Nations Development Programme (UNDP)* hat festgestellt, dass afghanische Sprachen und Schriften nicht in gängiger Software enthalten sind und hat daher deren Integration empfohlen (Pournader und Everson 2003). Eine Anpassung von Open-Source-Software an lokale Schriften, Sprachen und Gegebenheiten eröffnet eine Chance für die lokale Wirtschaft, welche die Entwicklung, die Installation und den Support des angepassten Systems vornehmen könnte (Ghosh 2004, Weber 2003, S. 17).

Die an lokale Sprachen angepasste Software kann genutzt werden, um Menschen den Zugang zu Computern zu ermöglichen, die über nicht ausreichende Englischkenntnisse verfügen. Dies spielt in Afghanistan eine große Rolle, da nur ein sehr kleiner Teil der Bevölkerung über mittelmäßige oder gute Englischkenntnisse verfügt. Den meisten Afghanen bleibt daher die umfassende Benutzung eines Computers allein aufgrund der Sprache verschlossen.

Trotz all dieser Gründe, die für eine Übersetzung von Open-Source-Software sprechen, wurde bisher keine Übersetzung von Open-Source-Software in eine der Lan-

8 https://launchpad.net/rosetta

dessprachen Afghanistans vorgenommen. Bei Betrachtung der zuvor beschriebenen mangelnden Verbreitung von Open Source in Afghanistan mag dieser Umstand nicht verwunderlich erscheinen. Microsoft arbeitet mit dem afghanischen Ministerium für Telekommunikation zusammen, um Windows Vista und Office nach *Dari*, welches die primäre Landessprache in Afghanistan ist, zu übersetzen. Es existiert hingegen einige Open-Source-Software auf Persisch, der Landessprache Irans, die große Ähnlichkeit mit *Dari*aufweist.[9]

5.2 Dateiformate, Protokolle und Kompatibilität

Standards legen die Art und Weise, in der die Kommunikation zwischen Systemen und auch die Speicherung von Daten durchgeführt wird, fest. Proprietäre Standards werden von einzelnen Firmen definiert, weiterentwickelt und häufig nicht offengelegt. Ein Beispiel ist Microsofts DOC-Format.

Offene Standards werden hingegen häufig durch unabhängige Organisationen definiert oder entwickeln sich aus universitären sowie Open-Source-Projekten. Die Spezifikationen offener Standards sind öffentlich zugänglich, so dass sie jeder interessierte Entwickler in seinem System verwenden kann. So ist es möglich, eine Vielzahl von Implementierungen zu erreichen, die aufgrund der Verwendung der gleichen Standards kompatibel zueinander sind. Populäre Beispiele sind E-Mail, *World Wide Web* und neuerdings *OpenDocument*, welches u. a. von *OpenOffice.org* verwendet wird. Die Benutzung proprietärer Formate erschwert in Folge von fehlender oder mangelhafter Kompatibilität die Einführung alternativer Systeme, welche mit den proprietären Systemen kommunizieren oder in proprietären Datenformaten erstellte Dateien verwenden, wie z. B. Open-Source-Software (Weber 2003, S. 18).

Wie bereits beschrieben, musste bei unseren Projekten keine bestehende IT berücksichtigt werden. Stattdessen konnten wir das Konzept auf der „grünen Wiese" planen und implementieren. Im ITCK stellten sich im späteren Verlauf insbesondere zwei Abhängigkeiten heraus: Zum einen dominiert das DOC-Format beim Austausch von Textdokumenten sowohl innerhalb des ITCK als auch mit externen Instanzen. Die Kompatibilität mit *OpenOffice.org* oder anderen Open-Source-Textverarbeitungen ist häufig ausreichend, aber im Detail bzw. bei großen Dateien fehlerbehaftet. Zum anderen werden die PCs für spezielle Applikationen, wie z. B. *AutoCAD*, *Photoshop* und *Dreamweaver* verwendet, die nur unter Windows lauffähig sind. Es bleibt offen, ob die Verwendung dieser Applikationen mit Hilfe von *Wine*[10] unter Linux praktikabel ist und ob alternative Open-Source-Anwendungen ausreichend leistungsfähig und benutzerfreundlich wären, um auf Akzeptanz zu stoßen.

Die in unseren Projekten am weitesten verbreiteten Netzwerkprotokolle sind *TCP*, *IP*, *SMB/CIFS*, *LDAP* und diverse Instant-Messenger, die entweder auf offenen

9 Siehe http://www.parsix.org.

10 http://winehq.org

Standards beruhen oder unter Linux ausreichend implementiert sind (vgl. Cerri und Fuggetta 2007; Krechner 2005). Es bestand daher aufgrund der Netzwerkprotokolle, anders als bei Dateiformaten, kein Anreiz, Windows-Systeme zu verwenden.

6 Ausblick

Der vorliegende Text stellt einen Erfahrungsbericht dar und daher sind die daraus zu gewinnenden Erkenntnisse nur eingeschränkt verallgemeinerbar. Offen bleibt, wie das Ergebnis einer empirischen Untersuchung ausfallen würde, bei der u. a. mit Kontrollgruppen – etwa in Form von Studentengruppen, die nur auf proprietärer Software ausgebildet werden – gearbeitet wird. Weiterhin wäre zu untersuchen, inwiefern sich die vorliegenden Ergebnisse auf andere sich entwickelnde Länder übertragen lassen. Es bleibt jedenfalls festzuhalten, dass Open Source mit all seinen Freiheiten zumindest die Entwicklung der Studierenden während ihrer Ausbildung gefördert hat und dass dadurch ein optimaleres Lernverhalten bezüglich der Zusammenhänge der Informationstechnologie bei ihnen verankert werden konnte.

Es ist nun die Aufgabe der Politik, Bildung als Grundlage für die Identitäts- und Persönlichkeitsentwicklung in Afghanistan zu erklären und dabei besonders auch Konzepte wie Open Source zu berücksichtigen (vgl. Barton et al. 2002). Bildung sollte nach Meinung der Verfasser Übersichtswissen, strukturelles Denken und Wertmaßstäbe in einem Gesamtkonzept vereinigen. Zwar haben die Menschen in Afghanistan, mit denen wir Kontakt hatten, meist eine Vorstellung davon, was sie technisch umsetzen wollen, aber ihnen fehlt das erforderliche Hintergrundwissen, das sich die Menschen in entwickelten Ländern im Laufe der durchlebten technischen Entwicklung aneignen konnten. Außerdem fehlt ihnen die notwendige IT-Kompetenz. Gerade deswegen sollte die Bildung und Ausbildung zum wichtigsten Ziel der Entwicklungszusammenarbeit in Afghanistan werden.

Es sollte eine Bildungsreform für Afghanistan entwickelt und umgesetzt werden, die zeitgemäß, kostengünstig und vor allem nachhaltig ist. Das heißt organisatorisch und inhaltlich sollte das Bildungswesen neu gestaltet werden, wobei Open Source eine große Rolle spielen kann. Mit Open-Source-Software hat sich eine neue, praxistaugliche Alternative auf dem Gebiet des Softwaremarktes entwickelt. Hierdurch wird der Wettbewerb mit herkömmlichen Lizenzmodellen verstärkt und die Abhängigkeit von einzelnen Herstellern verringert. Die afghanische Regierung wäre gut beraten, künftig den Einsatz von Open-Source-Software in der Wissenschaft und in der Verwaltung zu unterstützen und die Menschen über bestehende Software-Alternativen im Bereich Open Source zu informieren.

Open Source bietet eine große Chance für den Aufbau der akademischen Strukturen in Afghanistan (vgl. Peroz 2000, 2001*a*,*b*). Da das Land nur über wenig Ressourcen verfügt und entsprechend im Vergleich mit anderen Ländern nicht so viel in die Bildung investieren kann, bietet gerade diese preiswerte, zukunftssichere und in Bezug auf die

Hardware auch meist recht genügsame Software die Möglichkeit, mit wenig Mitteln mehr Menschen zu erreichen.

Literatur

Barton, J., Alexander, D., Correa, C., Mashelkar, R., Samuels, G. und Thomas, S. (2002), Integrating Intellectual Property Rights and Development: Policy Copyright, Software and The Internet, Final Report, Commission on Intellectual Property Rights, London. http://www.iprcommission.org/papers/pdfs/final_report/Ch5final.pdf [05. Dez 2006].

Becta (2005), Open Source Software in Schools, Projektbericht, British Educational Communications and Technology Agency. http://www.becta.org.uk/corporate/publications/documents/BEC5606_Full_report18.pdf [12. Dez 2006].

Bozman, J., Gillen, A., Kolodgy, C., Kusnetzky, D., Perry, R. und Shiang, D. (2002), Windows 2000 Versus Linux in Enterprise Computing, IDC White Paper, IDC. http://www.microsoft.com/windows2000/docs/TCO.pdf [27. Jan 2007].

Cerri, D. und Fuggetta, A. (2007), Open Standards, Open Formats, and Open Source, Working Paper, CEFRIEL - Politechnico di Milano, Milan. http://alfonsofuggetta.org/mambo/images/stories/Documents/Papers/openness.pdf [19. Feb 2007].

Deutscher Bundestag (2001), 'Beteiligung bewaffneter deutscher Streitkräfte an dem Einsatz einer Internationalen Sicherheitsunterstützungstruppe in Afghanistan auf Grundlage der Resolutionen 1386 (2001), 1383 (2001) und 1378 (2001) des Sicherheitsrats der Vereinten Nationen', Antrag der Bundesregierung, Drucksache 14/7930. http://www.auswaertiges-amt.de/diplo/de/Aussenpolitik/RegionaleSchwerpunkte/Afghanistan/BTDrucksache1407930.pdf [03. Feb 2007].

Drakos, N., Di Maio, A. und Simpson, R. (2003), Open-Source Software Running for Public Office, Studie, Gartner Inc. http://www.gartner.com/DisplayDocument?ref=g_search id=392799 [30. Nov 2006].

Ghosh, R. A. (2003), 'License Fees and GDP per Capita: The Case for Open Source in Developing Countries', *First Monday* **8**(12). http://firstmonday.org/issues/issue8_12/ghosh/index.html [30. Nov 2006].

Ghosh, R. A. (2004), The Opportunities of Free/Libre/Open Source Software for Developing Countries, *in* 'UNCTAD-ICTSD Dialogue on Moving the Pro-Development IP Agenda Forward: Preserving Public Goods in Health, Education and Learning', Bellagio. http://www.iprsonline.org/unctadictsd/bellagio/docs/Gosh_Bellagio4.pdf [30. Nov 2006].

Höhn, S. und Herr, G. (2004), Open Source: Open for Business, Studie, Leading Edge Forum. http://www.csc.com/features/2004/uploads/LEF_OPENSOURCE.pdf [12. Dez 2006].

Krechner, K. (2005), Open Standards Requirements, *in* 'Proceedings of the 38th Annual Hawaii International Conference on System Sciences'. http://www.csrstds.com/openstds.pdf [26. Sep 2006].

Krotoski, A. (2005), 'Programmierer als Entwicklungshelfer', Technology Review. http://www.heise.de/tr/aktuell/meldung/56499 [30. Nov 2006].

Mahr, B., Peroz, N., Balschun, A., Hoffmann, E. M., Suhr, J. und Tippmann, D. (2006), *Establishing Academic Structures in Computer Science at Herat University*, IKO, Frankfurt a. M.

Peroz, N. (2000), 'Nationale Konzepte', *c't – Magazin für Computertechnik* **10/2000**, S. 100. Interview.

Peroz, N. (2001*a*), Internet in Afghanistan, *in* J. Becker und S. Ramanathan (Hrsg.), 'Internet in Asia', Asian Media Information and Communication Centre, Singapur.

Peroz, N. (2001*b*), 'Mit wenig Mitteln viele Menschen erreichen', *c't – Magazin für Computertechnik* **10/2001**, S. 106. Interview.

Pournader, R. und Everson, M. (2003), 'Computer Locale Requirements for Afghanistan', United Nations Development Programme (UNDP). Interview, http://www.evertype.com/standards/af/af-locales.pdf [30. Nov 2006].

Rajani, N., Rekola, J. und Mielonen, T. (2003), Free as in Education. Significance of the Free/Libre and Open Source Software for Developing Countries, *in* 'World Summit on the Information Society (WSIS)', Genf. http://www.eldis.org/static/DOC12694.htm [30. Nov 2006].

Robert Francis Group (2002), Total Cost of Ownership for Linux in the Enterprise, Studie, Robert Francis Group. http://www.ibm.com/linux/RFG-LinuxTCO-vFINAL-Jul2002.pdf [27. Jan 2007].

UNCTAD secretariat (2003), 'E-Commerce and Development Report', United Nations Conference on Trade and Development. http://www.unctad.org/en/docs/ecdr2003ch4_en.pdf [30. Nov 2006].

Weber, S. (2003), Open Source Software in Developing Economies, Working Paper, University of California, Berkeley. http://www.ssrc.org/programs/itic/publications/ITST_materials/webernote2.pdf [30. Nov 2006].

Wheeler, D. A. (2005), 'Why Open Source Software / Free Software (OSS/FS)? Look at the Numbers!'. http://www.dwheeler.com/oss_fs_why.html [30. Nov 2006].

Kapitel 6

Open Access

„Der Denker und ebenso der Künstler, welcher sein besseres Selbst in Werke geflüchtet hat, empfindet eine fast boshafte Freude, wenn er sieht, wie sein Leib und Geist langsam von der Zeit angebrochen und zerstört werden, als ob er aus einem Winkel einen Dieb an seinem Geldschranke arbeiten sähe, während er weiß, daß dieser leer ist und alle Schätze gerettet sind."

– Friedrich Nietzsche

Kinderhände

GERHARD SCHÖPPE

(CC-Lizenz siehe Seite 563)

Der Antrag auf Einreise in die DDR wurde dann ausgefüllt, wenn der West-Berliner sich eigene Bücher gewisser Autoren verschaffen wollte, doch knapp bei Kasse war. Literatur auszuführen war legal; auffällig war es nur, mehr Wert in Mark an Literatur an der Grenze dabeizuhaben, als auf der Umtauschquittung zum Beleg des Tausches von D-Mark in Mark der Deutschen Demokratischen Republik ausgewiesen war.

Heutzutage geht es noch viel billiger: Ein paar Tasten getippt, schon liefern Googleisomere den Pfad, und nach ein, zwei Klicks ist die gewünschte Information da. Das geht immer dann gut, wenn der Autor populär und möglichst lange tot ist und in der gewünschten Sprache auch geschrieben hat. Urheberrechte haben noch ein Verfallsdatum, danach ist der Text frei.

Es gibt auch hier noch Nebenbedingungen: Zuerst ist es notwendig, dass der Suchende lesen kann. Achthundertsechzig Millionen Menschen verfügen nicht über diese Fähigkeit. In Deutschland wird der Anteil der funktionalen Analphabeten bei sechs bis elf Prozent geschätzt. Weiterhin braucht der Sucher Zugang zum Internet, den nicht alle haben, die des Lesens fähig sind.

Sind nun all diese Voraussetzungen gegeben, ist auch der Server gefunden, auf dem die Information bereitsteht, so heißt es noch nicht, dass das Gewünschte nun genutzt werden kann. In Berliner Schulen gab es früher die Lehrmittelfreiheit – egal, welcher Herkunft das Kind war, die Bücher zum Unterricht gab es von der Schule – zumeist übereignet.

Hilf, Mantz, Dreier und Stockmar sowie Ishii haben in ihren folgenden Beiträgen weder zur DDR noch über Analphabeten geschrieben, doch über Open Access und Open Content. Ihre Beiträge eröffnen den Überblick, untersuchen die Theorie und berichten aus der Praxis des kostenfreien Zugriffs auf – hier werden die oben anklingenden Grundbedingungen des Mediums Internet, Medienverfügbarkeit und Medienkompetenz außen vor gelassen – im weiteren Sinne literarische Informationsgüter.

Die Autoren dieses Abschnitts machen den Leser mit der Problematik vertraut – wobei der Überblick zur Einordnung in das „Open"-Feld von Reto Mantz, die vertie-

fende Beschreibung zu Open Access von Eberhard Hilf geliefert wird. Und Kei Ishii schreibt aus der Praxis, den Nachlass seines Vaters, des Komponisten Maki Ishii, frei zugänglich zu machen. Sie stellen insgesamt Formen vor, in denen wissenschaftliche Arbeiten so publiziert werden, dass Autoren wie Leser daraus Gewinn ziehen können – im Sinne einer für beide Seiten fruchtbaren Partnerschaft, wie sie der Komponist Maki Ishii Zeit seines Lebens pflegte.

Der Gegenstand der Aufsätze beinhaltet die Erweiterung der Basis sprachlichen Denkens in die Wissenschaft und die Kultur der heutigen Zeit. Die Worte und Begiffe, die dem Menschen vertraut werden – wie unterschiedlich seine Umgebung auch sei, während er heranwächst –, prägen seine Sprache und so sein sprachliches Denken wie sein kulturelles Leben. In den weiteren Schritten seines Spracherwerbs greift der des Lesens kundige Mensch – so es ihm möglich ist und er aus Neigung oder durch verschiedenste Einflüsse dorthin bewegt wird – weiter aus, zum Geschriebenen.

Denn die Erinnnerung kennt bei den meisten Menschen Grenzen. So kam Schrift, später die Fotografie, der Film, das Tonband; sie wurden benutzt, um das Wissen der Menschen zu Menschen weiter – auch über die Zeit – zu tragen; damit dies Wissen über die Kapazität der Erinnerung des Einzelnen, auch seiner Gruppe, hinauswachsen konnte.

Digitalisierung der Informationsgüter hat in Verbindung mit dem Ausbau ihrer Übertragungswege den Erwachsenen eine Geste ihrer Kindheit zurückgegeben. All die Zeit, die ihre Mütter und Väter, wie sie selbst, allein damit verbringen mussten, in Bibliothekskatalogen mit ihren Händen zu suchen, Regale abzuschreiten, nur um in Literaturverzeichnissen des Gefundenen weitere Literatur auszumachen, die ebensolche Tätigkeit einschloss – all dies scheint obsolet zu werden. Nun könnten sie, wie in ihrer Kindheit, nun mit ihren Händen auf der Tastatur – nahezu schrankenlos – den riesigen Raum des Wissens ihrer Zeit- und Vorzeitgenossen erspüren, begreifen und aufnehmen.

Doch scheint es deutlich, dass nur wenn diesen Kindern unserer Zeit nicht die Hände gebunden werden, sie nicht zur Einsicht in jede Literatur erst erhebliche Beträge – wenn auch bequem über das Online-Bezahlsystem – zu zahlen haben, eine neue Qualität des Wissens – für viele, ohne Schranken – aufblühen und erwachsen werden wird. Doch auch diese Vorstellung ist, wie etwa im Beitrag von Dreier und Stockmar zur *Göttinger Erklärung* ausgeführt, nicht allgemein.

Der letzte Autor dieses Abschnitts nun wagt einen Blick voraus – nicht mit den Mitteln der Wissenschaft –, der Gewinner des Schreibwettbewerbs „Open Fiction" schreibt über die neuen Ossis.

Digitaler Open Access zu wissenschaftlichen Informationen – Ein Umbruch zu neuen professionellen Diensten

EBERHARD R. HILF*

(CC-Lizenz siehe Seite 563)

Wissenschaftliche Informationsbeschaffung im digitalen Zeitalter kann besonders effektiv unterstützt werden, wenn eine digitale Kopie möglichst aller wissenschaftlichen Dokumente Open Access (OA), also für jedermann frei zugänglich, im Web zu beziehen ist. Wir werden die vielfältigen Möglichkeiten und Angebote, Dokumente OA zu legen, daraufhin analysieren, welche Mehrwertdienste sie bieten und wie erfolgversprechend die *Policy* (langfristige Strategie) der anbietenden Institutionen und Dienstleister ist, OA flächendeckend durchzusetzen. Dabei wird sich eine potenziell herausragende Rolle der Informationsdienstleister in den Hochschulen und wissenschaftlichen Instituten herausstellen. Mit zu OA passenden Geschäftsmodellen bieten sich aber auch sehr gute Chancen für innovative Firmen, in der OA-Welt geeignete Mehrwert-Dienstleistungen anzubieten. Strukturell vollzieht sich ein dramatischer Wandel von einer in der Papierwelt etablierten und an sie angepassten festen Wertschöpfungskette in der Verlagswirtschaft, die zur derzeitigen Monopolstruktur geführt hatte, zu einem breiten Fächer unabhängiger digitaler Mehrwertdienste. Sie wird zu einer effektiveren Befriedigung der ungeänderten Anforderungen aus der Wissenschaft führen: bestmöglich, umfassend und zeitnah mit Informationen versorgt zu werden.

Schlüsselwörter: Open Archives Initiative · Open-Access-Geschäftsmodell · offenes Publizieren · HAL · ArXiV · Metadaten

* Mein Dank für gute Hinweise geht an Stevan Harnad, Alma Swan, Frank Laloé und Arthur Sale.

1 Was ist Open Access?

In der digital vernetzten Welt sind Open Source und Open Access verdeckt miteinander verwandt: Bei Open-Source-Software dürfen weltweit alle Programmierer, die an der Verbesserung, Erweiterung oder Korrektur eines Programm-Codes arbeiten wollen, dies barrierefrei und unmittelbar tun. Dadurch können in sehr kurzer Zeit von der Community gewünschte Erweiterungen oder Anpassungen geleistet sowie Fehler diskutiert, gefunden und ausgemerzt werden. Der technische Fortschritt kann so maximal gefördert werden. Der (u. U. katalytisch riesige) wirtschaftliche Mehrwert stellt sich über die freie und damit maximal mögliche Nutzung der Software ein.

In der wissenschaftlichen Welt nennt man digitale Dokumente Open Access (OA) gelegt, wenn sie online barrierefrei und kostenlos über das Internet gefunden, heruntergeladen und im Volltext gelesen werden können. Ist dagegen für den Zugang der Kauf einer (digitalen oder Papier-) Zeitschrift notwendig, was den Leserkreis naturgemäß einschränkt, oder kann der Zugang nur durch Sicht auf ein Terminal in einer Bibliothek erfolgen (Einschränkung der Nutzung: z. B. kein Download), so wird dies oft entsprechend als „Non-OA" oder „Toll Access" bezeichnet. Jeder Forscher weltweit, der ein Dokument für seine Arbeit braucht, kann dieses also lesen und verwenden, wenn es Open Access verfügbar ist. Der wissenschaftliche Fortschritt kann zeitlich unmittelbar und unbeschränkt durch maximal mögliche Informationsversorgung gefördert werden. Der eigentliche wirtschaftliche Mehrwert stellt sich indirekt ein: Erst aus der breitestmöglichen und zeitlich unmittelbaren *Nutzung* neuen Wissens durch alle interessierten Dritten aus Forschung und Entwicklung erwächst der wissenschaftlich-technologische Vorsprung, der wiederum die Wirtschaft antreibt.

Open Access zielt nur auf den allgemeinen Zugang zum Wissen, das in einem wissenschaftlichen Dokument mitgeteilt wird. Es lässt dem Autor bzw. seiner Institution die Freiheit, sich davor, parallel oder danach für eine Referierung, eine Vermarktung durch Aufnahme in einer wissenschaftlichen Zeitschrift oder Herausgabe als Buch in einem Verlag zu entscheiden. OA entspricht daher in erster Linie einer Willenserklärung des Autors, seine wissenschaftlichen Ergebnisse allen möglichen interessierten Lesern barrierefrei in einer digitalen Kopie online zugänglich zu machen. Es schränkt ihn nicht in seinem Recht auf wirtschaftliche Verwertung ein. Open-Access-Dokumente sind frei zugängliche digitale Kopien wissenschaftlicher Dokumente. Autoren können die von ihnen gewünschte Rechtevergabe auch selbst definieren.[1]

Dieses parallele digitale Open-Access-Legen eines Artikels erhöht wesentlich die Zahl derer, die diesen Artikel (z. B. über Suchmaschinen) finden können, ist also kostenloses Marketing für den Verlag. Ob dies dann auch für den Verlag zu mehr Bestellungen bzw. Anforderungen von Sonderdrucken führt, hängt von vielen Faktoren ab. So werden Druckaufträge umso häufiger erteilt und für den Verlag lukrativer, je größer die Seitenzahl des Dokumentes ist.

1 http://www.creativecommons.ch

2 Verbreitung und Zitationsrate von Open Access

Der größere wissenschaftliche *impact* – die Wirkung auf den Fortschritt der Wissenschaft – von OA-Dokumenten wird von der Wissenschaft der Bibliometrie versucht, mittels möglichst geeigneter Maße zu messen. In einer internationalen Studie (Harnad et al. 2004), die sich über eine größere Anzahl von Fächern erstreckte, wurde über die letzten zwölf Jahre die Zahl der Zitationen pro Dokument gemessen. Je nach Fach erwies sie sich als zwei- bis zehnmal höher als für Toll-Access-Dokumente derselben Disziplin im selben Jahr.

In derselben Studie wurde auch der Anteil der OA-Dokumente an der gesamten Menge der Dokumente eines Faches und Jahres gemessen. Er stieg seit 1991 stetig von etwa 2 Prozent auf etwa 15 Prozent – übrigens unabhängig davon, ob der jeweils entsprechende Verlag in seiner mit dem Autor vereinbarten Urheberrechtsverfügung dies erlaubt oder nicht, ein Hinweis darauf, dass den allermeisten Autoren gar nicht bewusst ist, was sie mit dem Einsenden ihres Dokumentes zur Publikation in einer von ihnen gewählten Zeitschrift mit dem entsprechenden Verlag vereinbart haben.

Die allermeisten Verlage (über 90 Prozent) erlauben jedoch dem Autor, sein Dokument unabhängig von der Zeitschriftenveröffentlichung digital selbst auf einen Server seiner Wahl zu legen. Die genauen Regeln sind in der so genannten Romeo/Sherpa–Liste zusammengestellt.[2]

Ein direkter Einblick in den Zusammenhang von Open Access und Zitationsrate (*impact*) ergibt sich aus der vom BMJ-Verlag (*British Medical Journal*) freundlicherweise veröffentlichten Statistik[3] der Zitationsraten seiner Publikationen und der Zugriffe auf seine Zeitschriften. 1998 wechselte der BMJ-Verlag sein Geschäftsmodell und legte alle seine Zeitschriften offen (Open Access). Seither stiegen die Downloadzahlen linear von unter 100 000 pro Woche auf über zwei Millionen. Der *impact* stieg von fünf auf acht Zitierungen je Artikel als Ergebnis der viel größeren Leserschar. Spannend wurde es nun, als Anfang 2005 der Verlag wieder gemäßigt zu Toll Access zurückkehrte, genauer: Die digitalen Artikel blieben weitgehend OA, aber alle Mehrwertdienste (Letters, News, Front-end-Material etc.) wurden Toll Access. Von diesem Zeitpunkt an gingen die Nutzungen stark zurück, und es bleibt abzuwarten, wie die Zitationszahlen reagieren werden. Für Strategien anderer Verlage gibt unter anderem *Essential Science Indicators* (Bodenschatz und Smith 2006) weitere Informationen.

Bisher beziehen sich die meisten Untersuchungen nur auf den Teilbereich der Publikationen in Zeitschriften. Jedoch sind derzeit Studien und Werkzeuge zur Zitationsauswertung von wissenschaftlichen Dokumenten, die in digitalen Archiven im Web vorhanden sind, in Vorbereitung.[4]

2 http://romeo.eprints.org

3 http://www.bmj.com/aboutsite/visitorstats.shtml

4 Z. B. bereitet *DINI*, die *Deutsche Initiative für NetzwerkInformation* das Projekt „Erschließung von Zitationen in verteilten Open Access Repositories“ vor.

3 Archive für Open-Access-Dokumente

Die verschiedenen gebräuchlichen Methoden, wissenschaftliche Dokumente OA zu legen, seien nun kurz in ihren Besonderheiten vorgestellt und dann analysiert.

Der Autor kann die digitale Kopie

- auf seinen eigenen Instituts-Webserver legen (*Personal Repository*, PR)[5] bzw. lässt dies durch den zuständigen Operator besorgen. Als *Repository* wird in der internationalen Diskussion dabei allgemein ein digitales Dokumenten-Archiv verstanden, dessen Dokumente durch ein Webinterface über das World Wide Web frei zugänglich sind. Für sein eigenes PR entscheidet der Autor allein, welche Dokumente er dort gespeichert haben möchte. Diese Auswahl ist i. A. identisch mit seinem Verzeichnis wissenschaftlicher Arbeiten (Publikationsverzeichnis).
- bei dem *Institutional Repository* (IR) seiner Institution, wie z. B. seiner Universität einreichen (Dobratz 2006). Die *Institutional Repositories* unterscheiden sich allgemein vor allem durch verlangte Rechteüberschreibungen (z. B. Erlaubnis oder Verbot der Zweitverwertung durch den Autor), durch die Art der betreibenden Institution (Wissenschaftsinstitution, Universität, Verlag usw.), durch die angebotenen Zusatzdienste (wie z. B. Langzeitarchivierung, Metadaten, semantische Dienste, Zitationsdatenbank etc.; siehe auch Hilf und Severiens 2006). Das IR entscheidet auf Grund seiner *Policy*, welche Dokumente es aufnimmt.
- einem *Central Repository* (CR) zusenden, z. B. der Deutschen Nationalbibliothek[6], oder einem der internationalen „eArchive", wie etwa dem bereits 1991 gegründeten *ArXiv*[7], dessen Umfang immer noch exponentiell wächst, oder dem in Frankreich neu gegründeten internationalen *HAL*.[8] Die Aufnahmekriterien beschränken sich meist auf technisch-formale Korrektheit des digitalen Files.
- einem Verlag senden, der seine Publikationen als *preprint* oder als Publikation (nach der Referierung) online Open Access zugänglich macht. Der Verlag akzeptiert ein Dokument nach einer Fachbegutachtung innerhalb seines für eine Zeitschrift gesteckten Fachgebietsrahmens.

Die Motivationen für Open Access sind dabei für die verschiedenen Akteure durchaus sehr unterschiedlich. Die Autoren wollen weitestmöglich gelesen und zitiert wer-

5 Ein Beispiel für ein *Personal Repository* ist die Publikationsliste der AG Clusterphysik der Carl von Ossietzky Universität Oldenburg: http://smallsystems.isn-oldenburg.de/publications/.

6 Der Archivserver *Info Deposit*: http://deposit.d-nb.de.

7 *ArXiv* ist ein Archiv mit ca. 330 000 wissenschaftlichen Dokumenten aus Physik, Biologie und Mathematik: http://arxiv.org.

8 Das zentrale Online-Archiv *HAL* für alle wissenschaftlichen Fächer aller Länder ist ein Dienst des *Centre pour la Communication Scientifique Directe (CCSD). Hyper Article on Line* ist zu finden unter http://hal.archives-ouvertes.fr/index.php?langue=en.

den (wissenschaftlicher *impact* und Wissenschaftskarriere). Fachgesellschaften wollen durch OA-Zeitschriften und -Server zum Fortschritt ihres Faches beitragen, Universitäten wollen mit ihren *Repositories* den Nachweis ihrer wissenschaftlichen Leistung im Wettbewerb zu anderen Hochschulen um Studenten, Wissenschaftler und Finanzmittel unterstützen. Kommerzielle OA-Verlage wollen als professionelle Dienstleister wirtschaftlich bestehen.

4 Skalierung des Wachstums

Zurzeit gibt es eine lebhafte internationale Debatte (Bosc et al. 2006), mit welcher Strategie sich Open Access am raschesten umfassend und weltweit durchsetzen und realisieren ließe. Daher beleuchten wir nun kurz die zurzeit parallel verfolgten Strategien.

4.1 Personal Repositories

Um die Entwicklung von OA bei den *Personal Repositories* (PR) zu analysieren, arbeiten wir an einer Studie (Harnad et al. 2007) mit. Sie soll den Anteil der Einträge in Publikationslisten von Autoren auf dem eigenen Server ermitteln, die zu einer OA-Fassung der Arbeit führen. Einige der sich aus den ersten Tests ergebenden Arbeitshypothesen seien hier vorweggenommen:

1. „Kulturelle Unterschiede" bei der Ablage digitaler Dokumente gibt es nicht nur fächerweise, ortsbezogen oder national, sondern insbesondere personenbezogen: Es gibt im selben Fachbereich vor allem einerseits Kollegen, die alle ihre Arbeiten OA legen (100 % OA), – entweder auf demselben Server wie die Publikationsliste (PR) oder aber in einem *Central Repository* (CR) (z. B. *SPIRES*), bisher aber nur selten auf einem *Institutional Repository* (IR) – andererseits Kollegen, die gar kein Dokument OA gelegt haben oder OA legen haben lassen. Die Mehrzahl der Autoren weisen in ihren Publikationslisten entweder so gut wie alle oder keine Arbeiten im Volltext nach.

2. Der Anteil der Autoren, die keine Publikationsliste online gelegt haben, variiert fachspezifisch. Teilweise wollen Autoren auch keine allgemein zugängliche Publikationsliste online, weil diese Teil der vertraulichen Vita ist.

3. Die mittlere Prozentzahl der Dokumente, die OA gelegt sind („OA/all"), ist zwar fachweise und ortsweise sehr verschieden, wächst aber im Mittel kaum mit der Zeit über die letzten Jahre. Die berühmten stets genannten 15 Prozent „OA/all" sind also ein wenig besagender Mittelwert (Harnad et al. 2004). Am OA-Verhalten des Einzelnen hat sich in den letzten Jahren wenig geändert.

4. Keineswegs immer sind die lokalen Universitäts-Repositories bekannt.

5. Die eigenen Rechte, die die Verlage den Autoren einräumen, ihr Dokument selbst digital OA zu legen, sind weitestgehend unbekannt.

Im Ganzen ergibt sich ein erstaunlich statisches Bild. Die meisten Autoren haben über die letzten Jahre ihre Strategie bzgl. OA weder überdacht noch angepasst noch kennen sie die ihnen gebotenen Möglichkeiten.

4.2 Institutional Repositories

Institutional Repositories entstehen zurzeit unabhängig an sehr vielen wissenschaftlichen Institutionen, insbesondere den Universitäten. Betreiber sind zumeist die lokalen Bibliotheken. Oft ist jedoch der Füllungsgrad ihrer *Repositories* noch gering. Das Sammelverzeichnis der registrierten OAI-PMH-Repositories[9] *ROAR* (Brody 2006), weist insgesamt weltweit zurzeit nur etwa zwei Millionen Dokumente nach.

Studien von Alma Swan (Swan und Brown 2005) für die englische Forschungsorganisation *JISC* und die Umfrage des Projektes *VERSIONS*[10] zeigen jedoch, dass über 80 Prozent der Autoren eine digitale Kopie ihrer Arbeiten OA legen würden, wenn ihre Institution es von ihnen fordern würde. Universitäten, die OA für ihre Wissenschaftler „verpflichtend" gemacht haben und die entsprechenden Dienste (IR) anbieten, haben in kürzester Zeit den Übergang geschafft. Herausragende Beispiele sind die *Universidade do Minho*[11] in Portugal, die *University of Southampton* oder die *University of Tasmania*[12].

In Deutschland hat *DINI* die Koordination übernommen und es gibt Hilfestellungen zur Verfahrensweise und zur Realisierung: Universitäten können sich um ein Zertifikat bewerben, welches die wesentlichen Anforderungen (Lesbarkeit, Authentizität, Integrität, Langzeitarchivierung, Suchbarkeit) institutionell und zeitlich unbefristet dem Autor und Leser gegenüber sicherstellt.

Die zahlreichen Absichtserklärungen wissenschaftlicher Institutionen und Verbände haben das öffentliche Bewusstsein für die OA-Entwicklung geschärft. Förderinstitutionen wie die DFG in Deutschland oder der *Wellcome Trust* in Großbritannien legen Autoren, die von ihnen Fördergelder erhalten, dringend nahe, ihre Kopien ihrer neuen wissenschaftlichen Dokumente in Open-Access-Repositories zu hinterlegen. Ziel der Förderinstitutionen ist, die Leistungsfähigkeit der Wissenschaft zu fördern. Diese Signale haben aber die Autoren in der Breite statistisch nur kaum messbar erreicht. Die konkrete *Policy* nationaler Förderinstitutionen ist durchaus verschieden.

9 Die *Open Archives Initiative (OAI)* hat das *OAI Protocol for Metadata Harvesting (OAI-PMH)* entwickelt, das zur Abfrage und Übertragung von Metadaten dient.

10 http://www.lse.ac.uk/library/versions/

11 http://www.campusvirtual.uminho.pt/Default.aspx?tabindex=1&tabid=4&pageid=3&lang=pt-PT

12 http://eprints.utas.edu.au

In Deutschland wird ein Netzwerk verteilter möglichst hochqualitativer *Repositories* angestrebt.[13] In Frankreich wurde ein *Central Repository*[14] eingerichtet.

Das Motiv für Hochschulen, die an ihnen entstandenen wissenschaftlichen Dokumente OA zu legen, ist es, ihre wissenschaftlichen Leistungen und deren *impact* zu erfassen und so im Wettbewerb unter den Hochschulen um Forscher, Studenten und Finanzmittel zu punkten sowie die interne Mittelaufteilung zu steuern. Sie setzen hier daher auch konkrete Anreize. So gab die Universität Minho finanzielle Anreize für diejenigen Fachbereiche, die zuerst umfassend ihre neuen Dokumente eingeliefert hatten.

4.3 Central Repositories

Bei den zentralen bzw. nationalen *Repositories* ergibt sich ein ganz anderes Bild: Das älteste zentrale *Repository ArXiv* zeigt erstaunlicherweise eine seit über 15 Jahren absolut gleichmäßige *lineare* Zunahme der Zahl der monatlich neu eingereichten Manuskripte (Submissionsrate). Jeden folgenden Monat werden etwa 22 Artikel mehr eingereicht als im Monat zuvor.[15] Autoren, die das *ArXiv* als *Repository* wählen, nutzen es i. A. für alle ihre Arbeiten. Der Zuwachs der Submissionsrate des *ArXiv* geschieht also im Wesentlichen durch das Hinzukommen neuer Autoren und die Ausdehnung auf neue benachbarte Gebiete. Dabei sind einige Fachgebiete bereits hochgesättigt, d. h. fast alle Autoren nutzen das *ArXiv* und die entsprechenden Monats-Submissionsraten werden für dieses nahezu konstant.

Wann also würden alle wissenschaftlichen Arbeiten im *ArXiv* nachgewiesen werden? Selbst bei konservativ angenommenen weltweit 5 000 Universitäten à 200 aktiven Wissenschaftlern mit einer neuen Publikation im Jahr und einem Anteil von 10 Prozent der (mathematisch-naturwissenschaftlichen) Fächer, für die andere gewachsene zentrale Archive wie *PLOS*, *BioMedCentral*, *HAL*, *REPeC* etc. nicht in Frage kommen, wäre eine Abdeckung von 50 Prozent aller neuen wissenschaftlichen Dokumente erst in etwa 360 Jahren erreicht. Die direkte Werbewirkung von Autor zu Autor ist also überraschend gering.

Eine etwas andere Strategie liegt dem international offenen zentralen Archiv *HAL* zugrunde, das 2003 mit der Aufnahme von Daten begonnen hatte. Die nationale Forschungsorganisation Frankreichs, *CNRS*, hat sich als *Policy* gesetzt, langfristig eine digitale Kopie aller neuen wissenschaftlichen Dokumente aus Frankreich dort zu archivieren. Das Tempo der Durchsetzung an den Instituten bestimmt dann das Wachstum. Zurzeit werden in *HAL* digitale Kopien von etwa 1 800 Dokumenten pro Monat eingeliefert mit einer Steigerung von etwa 100 pro Monat. Schon 2008 wird *HAL* vom Umfang her das *ArXiv* überholt haben. Für Frankreich ist bereits

13 Siehe http://www.dini.de/dini/arbeitsgruppe/arbeitsgruppe_details.php?ID=5.
14 *HAL* – siehe Fußnote 8.
15 http://arxiv.org/show_monthly_submissions

im übernächsten Jahr zu erwarten, dass jedenfalls aus Frankreich mehr als 50 Prozent aller neuen wissenschaftlichen Dokumente in einer Kopie in *HAL* im Volltext nachgewiesen werden. Eine nationale staatliche OA-Policy für wissenschaftliche Dokumente führt also rasch auch zu einer faktischen Durchsetzung von Open Access, auch ohne direkten Zwang (englisch: *mandating*), wie es aber einige Hochschulen (z. B. Southampton, UK; Minho, PT) praktizieren.

Analoge, gute Erfahrungen mit einer nationalen *Policy* wurden in Deutschland mit den Dissertationen gemacht.[16] Die nationale Strategie sowie ein dauerhaft angelegter und professioneller staatlicher Dienst (hier: die Deutsche Nationalbibliothek im Verbund mit den lokalen Universitätsbibliotheken) führte binnen weniger Jahre zur Durchsetzung von OA. Im Jahr 2005 wurden bereits 33 Prozent aller Dissertationen in Deutschland digital archiviert (Korb 2006). In den Fächern Mathematik, Physik und Linguistik wurden bereits etwa 50 Prozent aufgenommen.

4.4 Open-Access-Verlage

Bei den Open-Access-Verlagen gibt es ebenfalls noch keine verlässliche integrierte Übersicht über die Gesamtzahl der OA publizierten Dokumente. Die Zahl der OA-Verlage, ist bisher noch klein, wächst aber rasch. Die *Elektronische Zeitschriftenbibliothek Regensburg (EZB)* verzeichnet zurzeit etwa 14 000 OA-Zeitschriften, unter ihnen auch einige Periodika, Institutsreportreihen etc., die nicht alle kommerzielle referierte Zeitschriften sind.[17] Nur eine kleine Untermenge von den referierten OA-Zeitschriften, zzt. 2 440, sind im *Directory of Open Access Journals (DOAJ)*[18] registriert. Auch von diesen sind einige eher Prototypen.

Insgesamt gibt es je nach den Maßstäben, die man anlegt, sowie der Sammeleffizienz der Directory-Betreiber zwischen 35 534 (*EZB*) und 150 000 wissenschaftliche referierte Zeitschriften (sog. Ulrich-Liste[19]). Entsprechend beträgt der Anteil der OA- zu den Non-OA-Zeitschriften zwischen optimistischen 39 Prozent (*EZB*) und 1,4 Prozent (*DOAJ*).

Auch die externen Schätzungen über die Kosten für eine Open-Access-Zeitschrift je Artikel gehen weit auseinander. In der Verlagswirtschaft sind sie natürlich notwendigerweise Betriebsgeheimnis. Der unabhängige Report von Robert Terry (2005) für den *Wellcome Trust* nennt als Haupteinsparpotenziale beim Übergang von Toll Access zu Open Access den weitgehenden Wegfall der Kosten für Werbung, Vertrieb, physische Vorhaltung in Bibliotheken, Druckaufbereitung etc. Insgesamt rechnet er mit Einsparungen von etwa zwei Drittel der Kosten für eine reine Online-Zeitschrift.

16 Siehe *Dissertationen Online* (http://www.dissonline.de) der Deutschen Nationalbibliothek.

17 http://rzblx1.uni-regensburg.de/ezeit/fl.phtml?colors=7&lang=de&selected_colors%5B%5D=1&bibid=AAAAA

18 http://www.doaj.org

19 http://www.ulrichsweb.com/ulrichsweb/

Ein gutes Beispiel ist die große referierte Zeitschrift *ACP*[20], die gerade die Autorenkosten auf 20 € pro Seite gesenkt hat. Bei den aktuellen rein digitalen OA-Zeitschriften wird gegenwärtig mit vielen Finanzierungsmodellen experimentiert (siehe Neumann 2006 für die Zeitschrift *German Medical Science*).

Dagegen verfolgen die großen kommerziellen Verlage wie *Springer*, *Wiley*, *Elsevier* im Wesentlichen zurzeit noch eine andere Strategie: Das klassische Produkt der Print-Zeitschrift wird parallel zum Subskriptionsvertrieb gegen eine einmalige Autorengebühr zusätzlich digital Open Access gestellt. Als Preis werden dann die Vollkosten einer traditionellen Zeitschrift pro Artikel in Rechnung gestellt, für die um die 3 000 € genannt werden.

4.5 Strategieauswahl

In jedem der OA-Repositories wird das Hauptziel, das Dokument für den offenen Zugriff zur Verfügung zu stellen, erreicht. Sie unterscheiden sich aber darin, wer die Arbeit macht, wie leicht das Dokument gefunden werden kann (*retrieval*), welche Zusatzdienste geboten werden sowie im Geschäftsmodell, in den Kosten und wer diese trägt.

5 Retrieval wissenschaftlicher Dokumente

Im Ozean der unzähligen Dokumente des World Wide Web stellen die wissenschaftlichen Arbeiten nur eine verschwindend kleine Minderheit dar. Selbst die Open Access gelegten Dokumente zu suchen und zu finden, wird eine immer größere Herausforderung. Die Wissenschaftler wollen und sollen gelesen (und zitiert) werden und nicht nur passiv ihre Produktivität dokumentieren.

Zwei verwandte Strategien haben sich weitgehend durchgesetzt, um die Sichtbarkeit wissenschaftlicher Werke als einen essentiellen Mehrwertdienst zu erhöhen und so den möglichen *impact* auf die Wissenschaft zu realisieren.

5.1 Passives Anbieten von Metadaten

Der *Data Provider* des digitalen Dokuments, der also das Dokument im World Wide Web anbietet, fügt Metadaten hinzu. Metadaten sind einfache inhaltliche (semantische) Aussagen über das Dokument, wie zum Beispiel Name und Adresse des Autors, Titel des Dokumentes, Publikationsort, Zeitschrift etc., die in einer international vereinbarten Sprache kodiert werden.

Von der *Dublin Core Metadata Initiative (DC)*[21] wurde eine Sprache geschaffen, in der jeder seine eigene *Metadata Profile Application Definition* formulieren und

20 *ACP* steht für *Atmospheric Chemistry and Physics*: http://www.copernicus.org/EGU/acp/.
21 http://www.dublincore.org

bei *DC* registrieren und damit international (vor allem den Suchmaschinenbetreibern) bekannt machen kann.

Es lassen sich also spezifische Profile von Metadatensätzen (z. B. verschiedene semantische Tiefe: von nur bibliographischen Daten bis zu mathematisch-algebraischen Eigenschaften oder Fachspezifika) kodieren. Zumeist wird aber ein Minimalsatz verwendet („DC-unqualified").

Auf Grund der bei *DC* hinterlegten Metadata-Profile können Suchmaschinen nun das Dokument insoweit „verstehen" und einordnen/ranken. Die Erfolge der Hervorhebung eines relevanten Dokuments aus dem Meer der irrelevanten Dokumente etwa in *Google* sind wirklich beeindruckend.[22]

5.2 Aktives Vermitteln von Metadaten

Allerdings bleibt die Intelligenz des Lesers hilfreich, durch genügend durchdachte Suchanfragen das Dokument aus dem Meer der irrelevanten präzise herauszuschälen. Fachspezifische Suchmaschinen versuchen daher zunehmend, das Web vorzufiltern.

Darauf aufbauend hat sich die *Open Archives Initiative (OAI)* gebildet.[23] Sie setzt auf *aktive* Zusammenstellungen der relevanten Arbeiten in bei ihr registrierten *Service Providern*, d. h. sie vermittelt nur Suchergebnisse durch Suche über *Data Provider*, die sich gewissen Minimalregeln unterwerfen. Diese erfolgreiche Strategie setzt sich bei den wissenschaftlichen Dokumenten derzeit durch: Es gibt einfach zu installierende Open-Source-Software, um selbst einen OAI-kompatiblen *Data* oder *Service Provider* aufzusetzen.[24] Immer mehr Universitäten und wissenschaftliche Institute installieren ihre *Repositories* als *OAI Data Provider* für die Arbeiten von Autoren aus ihrem Hause.[25]

6 Professionalisierung und Arbeitsteilung

Um die Chancen der Open-Access-Welt für neue Dienste für die wissenschaftliche Arbeit auszuschöpfen, bedarf es professioneller Dienstleister. Andererseits sind die einzelnen Dienste nicht mehr wie im Papier-Zeitalter in eine technisch bedingte Wertschöpfungskette eingezwängt, sondern können unabhängig entwickelt und angeboten werden. Es könnten sich neue Arbeitsteilungen und andere Anbieter ergeben als bisher. Wer also von den möglichen Akteuren hat die Motivation, einen bestimmten Teildienst anzubieten, und wie professionell wird er dies leisten können?

Der Autor Ein Dokument im PDF-Format auf den eigenen Server zu legen, ist für niemanden mehr ein Problem, soweit seine Institution dies erlaubt (und er

22 So wird etwa die Homepage des Autors in *Google* trotz der Konkurrenz von 5 000 000 Fundstellen prominent gefunden.

23 http://www.openarchives.org

24 Siehe *EPrints Free Software*: http://www.eprints.org/software.

25 Siehe http://www.openarchives.org/Register/BrowseSites für registrierte *Data Provider*.

möglicherweise die technische Hilfe des Operators in Anspruch nimmt). DC–Metadaten hinzuzufügen auch nicht, seit es dafür geeignete Web-Formulare gibt.[26] Aber will oder kann der Autor seinen Server so lange Zeit betreiben, wie er das Dokument gerne OA hätte? Erlaubt es ihm seine Institution, etwa wenn es sich um industrienahe angewandte Forschung handelt?

Die lokale Bibliothek Immerwährende Dienstaufgabe der Universitätsbibliotheken ist die lokale Bereitstellung von Informationen aus der Wissenschaft. Sie werden daher in der digitalen Welt auch für das *retrieval* von Informationen verantwortlich. Im Geben und Nehmen zu anderen Bibliotheken in der Verbreitung der wissenschaftlichen Ergebnisse aus der eigenen Universität, sich in das Anbieten entsprechender Dienste einarbeiten. Sehr viele Universitäten bieten ihren Autoren bereits ein IR (*Institutional Repository*). Da die Bibliotheken Dauereinrichtungen sind, haben sie die Möglichkeit, digitale Archive professionell und auch langfristiger zu pflegen. Eine noch unbewältigte Herausforderung ist zurzeit, dass die Autoren am Orte oft von dieser Alternative noch gar nichts wissen, ihr nicht trauen oder sich selbst vorbehalten möchten, was OA gelegt werden soll und was nicht. Zugleich haben aber die Universitäten eine starke Motivation, durch ein gutes eigenes IR die wissenschaftliche Sichtbarkeit ihrer Institution zu fördern und dies im Wettbewerb mit anderen Hochschulen und bei der Mitteleinwerbung zu vermarkten. Die lokalen Institutionsbibliotheken, die ein OA-Repository einrichten, haben ihre eigene Motivation – die Leistungsfähigkeit der Institution zu zeigen, der Arbeit der eigenen Wissenschaftler zu dienen. Sie können einen engen Kontakt zu den lokalen Autoren pflegen und sie bei der Erstellung der digitalen Fassung beraten. Zugleich skaliert diese Kooperation mit dem Wachsen des Internets und der Wissenschaft.

Die zentralen Archive Diese haben als Vorteile, dass sie echte Langzeitarchivierung anbieten können, dass sie – dank ihrer eigenen Forschungsabteilung – mit der raschen technischen Entwicklung Schritt halten können und dass sie auch sehr aufwendige semantische Dienste anbieten könnten.

Die Open-Access-Verlage Erst wenige OA-Zeitschriften haben ein überzeugendes Geschäftsmodell gefunden, mit dem sie sich finanzieren können (z. B. Autorengebühren, Zusatzdienste wie *printing on demand*, Werbung, Sponsoren, Fachgesellschaften etc.). Die für ein professionelles und kostengünstiges Arbeiten notwendige technische Vernetzung zu technischen Dienstleistern und Entwicklern sowie die Vernetzung der kleinen und mittleren Unternehmen untereinander ist noch gar nicht ausgeschöpft.

26 Z. B. *My Meta Maker*: http://www.isn-oldenburg.de/services/mmm/.

7 Informationsquellen, Diskussionen & Rechte zu Open Access

In Deutschland bietet *DINI*[27] Empfehlungen, Erfahrungsworkshops und Materialien an. Die Site *Zugang zum Wissen*[28] führt zu den wichtigsten einschlägigen Quellen, so z. B. zu den stets aktuellen *Open Access News* von Peter Suber oder den umfangreichen Diensten, Handreichungen und FAQs zu Open Access von Stevan Harnad und Mitarbeitern in Southampton oder dem aktuellen Diskussionsforum des *American Scientist*. Ein anderes Beispiel ist die Sammlung von OA-Quellen durch die Universität Bremen.[29]

Es gibt zurzeit eine lebhafte wissenschaftspolitische internationale Diskussion, ob und inwieweit OA verpflichtend werden soll, also, ob eine Institution, z. B. eine Universität, von ihren Autoren verlangen kann bzw. soll, dass diese – unabhängig von der weiteren Verwendung – eine digitale Kopie auf dem institutionseigenen *Data Provider* ablegen. Bei Dissertationen, die in der Universität entstehen, wurde frühzeitig (1996), angestoßen durch ein fächerübergreifendes DFG-Projekt der *Initiative Information und wissenschaftliche Kommunikation der wissenschaftlichen Fachgesellschaften (IuK)*, ein pragmatischer Weg gefunden: Im Wesentlichen reicht der Doktorand nach Abschluss der Promotion eine digitale Fassung bei der lokalen Universitätsbibliothek ein. Diese legt den Text OA auf ihren Server und sendet eine Kopie an die Deutsche Nationalbibliothek. Der Kandidat erhält dann die Promotionsurkunde.[30]

Das vieldiskutierte heikle Problem der Diplom- und anderen Examensarbeiten klammern wir hier aus. Einerseits „gehört" die Arbeit (nicht als Examensarbeit deklariert) dem Kandidaten, andererseits verlangen viele Universitäten, Fachbereiche und Betreuer das Recht, Examensarbeiten nicht auf den OA-Server ihrer Universität legen zu *müssen*.

8 Geschäftsmodelle

In der Papierwelt hatte sich für das wissenschaftliche Publizieren innerhalb von 140 Jahren ein hochprofessionelles differenziertes Realisierungsverfahren und Geschäftsmodell entwickelt, das allen beteiligten Akteuren – Autoren, Lesern, Bibliotheken, Gutachtern, Herausgebern etc. – vertraut ist.

Der Autor schreibt ein Manuskript (jeweils mit gegebener Umfangsbeschränkung), sendet es an eine ihm hierfür fachlich passende referierende Zeitschrift, deren Verlag editiert es, lässt es begutachten, stellt es mit themenverwandten anderen Manuskripten zu einem Zeitschriftenband zusammen (ohne den Autor zu fragen und ohne dem

27 Die *Deutsche Initiative für NetzwerkInformation e. V.* ist eine Vereinigung der Informationsdienstleister an Hochschulen und wissenschaftlichen Institutionen (Bibliotheken, Rechenzentren, Multimediazentren), siehe auch http://www.dini.de.

28 http://www.zugang-zum-wissen.de

29 http://elib.suub.uni-bremen.de/open_access.html

30 http://www.dissonline.de

Leser eine Wahl zu lassen), druckt diesen und vertreibt ihn über Grossisten, bis er im Regal der Institutsbibliothek vorrätig gehalten wird. Der einzelne Wissenschaftler versucht dann, für ihn relevante Artikel über Suchmaschinen oder die eigene Bibliothek zu finden. Allerdings: Diese hat von allen fachlich für die lokalen Wissenschaftsgruppen relevanten Zeitschriften nur noch einen sehr kleinen Teil (selbst Harvard als größte Universitätsbibliothek mit dem Anspruch „alles relevante" zu abonnieren, kann hiervon nur 40 Prozent bezahlen).

Die notwendigen Investitionen und Aufwendungen für diese komplexe, zeitaufwendige, durch technische Beschränkungen eingeengte Transferkette von Wissen wurden durch ein auf die technisch- und zeitaufwendige Produktion passendes rigides Geschäftsmodell abgesichert: Die Institution des potenziellen Lesers zahlt für einen langjährigen Dauer- und Pauschal-Vertrag. Da die Artikel nur auf diesem Wege zugänglich waren, ergab sich eine Monopolstellung auf den Besitz des Wissens durch den jeweiligen Verlag. In der Softwarewelt hatten sich zunächst Firmen durchgesetzt, die von der Verknappung des Zugangs zum Code durch Preisbarrieren leben – die größte Firma bildete sich sogar durch die geniale Geschäftsidee, Textdokumente, auch wenn sie als *ASCII* erstellt worden waren, vom Autor verschlüsseln zu lassen, verschlüsselt zum Leser zu transportieren – und sich das von beiden Seiten bezahlen zu lassen (*Microsoft Word*). Entsprechend haben sich in der Welt der wissenschaftlichen Information zunächst wenige (und miteinander verflochtene) Firmen durchgesetzt, die von der Verknappung und der Qualitätssteigerung (Begutachten, Editieren, Drucken, Archivieren usw.) des Wissens leben. In der digitalen Welt ist das OA-Legen von Dokumenten ohne merklichen Zeit- und Kostenaufwand leicht möglich, und damit ist dieser Mehrwert weltweit unmittelbar erschließbar. Die bisherigen technischen Randbedingungen der Papierwelt entfallen. Zusatzdienste, die bisher in der Papierwelt einen festen Platz in der Wertschöpfungskette haben mussten, lassen sich nun einzeln, unabhängig und im Wettbewerb von entsprechend innovativen und wagemutigen Dienstleistern professionell und an die Anforderungen optimal angepasst wahrnehmen. Die Wissenschaft sollte so optimal gefördert werden können. Die Forderungen der Wissenschaft nach bequemer, vollständiger, nachnutzbarer Information ohne Zeitverzug ließen sich nun realisieren. Daher sind neue Geschäftsmodelle gefragt,[31] die in dem digitalen Szenario die Finanzierung professioneller Einzeldienste im Wettbewerb tragen. Hier hat jeder Dienstleister sein eigenes Konzept: Institutionelle Dienstleister, wie z. B. Universitätsbibliotheken, schöpfen aus ihren staatlichen Mitteln, ihre Institutionen erhalten aber als Gegenwert die Steigerung der eigenen wissenschaftlichen Leistungsfähigkeit, Ansehen etc.

Es gibt bereits einige kommerzielle Dienstleister im Open-Access-Szenario, die die für sie arbeitsfreie maximale weltweite Verbreitung der Texte nutzen. Sie berechnen

31 Gleichwohl versuchen die Großverlage derzeit, das Urheberrecht so ändern zu lassen, dass es das alte Geschäftsmodell längstmöglich nutzbar hält, trotz der heftigen Gegenwehr aller Wissenschaftsorganisationen (Kuhlen et al. 2004) und vieler tausender Persönlichkeiten in Bildung und Wissenschaft.

z. T. den Nutzern die Leistung ihrer Zusatzdienste (Referieren, *printing on demand*, Verlinkung, Suche nach ähnlichen Dokumenten usw.).

Ohne einfache Antwort ist die Frage, ob das Open-Access-Legen von wissenschaftlichen Zeitschriften über den viel größeren Leserkreis sich als fast kostenloses Marketing nutzen lässt, d. h. ob es mehr oder weniger Bestellungen für die kostenpflichtigen Add-on-Produkte wie zum Beispiel das gedruckte Werk gibt. Ein interessantes Beispiel ist *Medknow Publications*[32] (Mumbai, Indien). Als es seine bis dahin kostenpflichtigen Subskriptionszeitschriften Open Access legte, stiegen die Zugriffe dramatisch, aber es gab auch aus aller Welt eine Zunahme der nach wie vor kostenpflichtigen Subskriptionen für die Druckfassungen, und der *impact*, die Zitationen je Artikel, stiegen im gegebenen Beispiel um den Faktor fünf.

Man hätte erwartet, dass gerade kleine und mittlere Verlage in breiter Front die neuen zukunftsorientierten Marktchancen ausloten, dabei vielleicht technisch anspruchsvolle Aufgaben outsourcen und wettbewerblich professionelle digitale Leistungen anbieten. Stattdessen leben wir noch weitgehend in einer Art Interregnum, in der wenige große Firmen, die die Wissensverknappung als Geschäftsmodell haben, noch exzellente Gewinne (z. T. oberhalb von 30 Prozent des Umsatzes) machen, auf der anderen Seite aber sich rasch und an vielen Stellen in der wissenschaftlichen Welt OA-Server sehr unterschiedlicher Qualität bilden, und zwar national und international. Eine solide, aufgabenbegründete, professionelle Arbeitsteilung der Beteiligten ist das noch nicht.

9 Die weitere Entwicklung

Zunächst einmal ist ja nicht zwingend, dass die bisher von der Universität auf Grund der technischen Notwendigkeiten in der Papierära zur Realisierung an Verlage ausgelagerten Glieder des Herstellungs- und Vertriebsweges weiter ausgelagert bleiben müssen. In der Papierwelt behielt die Universität nur die Erstellung des Textes, die Begutachtung und das Vorhalten der gedruckten Form. Einzelne Glieder der Wertschöpfungskette als Institution selbst anzubieten, kann nicht nur wirtschaftlichen, finanziellen Nöten entspringen, sondern bedient auch handfeste wissenschaftspolitische Interessen; sei es, dass die Universität selbst festlegen will, nach welchen Kriterien sie Arbeiten in ihren IR (*Data Provider*) aufnimmt, um ihr Profil zu definieren; sei es, dass die Langzeitarchivierung vom Staat selbst als kulturpolitische Aufgabe (Severiens und Hilf 2006) wahrgenommen werden will oder dass die Anforderungen der Wissenschaftler an Literaturversorgung und Verbreitung der eigenen Werke am ehesten von der eigenen Bibliothek und in direktem Kontakt bedient werden können.

Die Entwicklung bei Open Source ist der von OA voraus, sodass sich für uns daraus lernen lässt. Jüngst wurde bei hier diskutiert, dass der auch breite, finanziell barrierefreie Zugang aller weltweit, auch aus den Entwicklungsländern, für die wirtschaftliche

32 Vergleiche Sahu et al. (2005).

Entwicklung beider Seiten von Vorteil ist. Entsprechendes gilt für den ungehinderten freien Zugang zu wissenschaftlicher Information: Die Weltbehörden[33] fordern es aus gleichem Grund. Auch hier ist klar, dass die Realisierung finanziell leicht machbar ist, wenn der Zugang OA ist und in der wissenschaftlichen Welt durch die Institutionen von ihren Autoren verlangt wird, eine digitale Kopie ihrer Dokumente OA zu legen, unabhängig von einer kommerziellen Zweitverwertung durch den Autor.

Wenn so klar und offensichtlich ist, was zu tun ist, warum wird es dann nicht von den reichen Industrieländern unmittelbar und ohne Zeitvertun umgesetzt? Hier kommen zwei verdeckte Mechanismen zum Tragen, die im Folgenden benannt werden sollen:

1. Der Übergang vom Dokumentenmanagement in der Papierära zur digitalen Ära zeigt auch Eigenschaften, die für Systemübergänge gelten. Es gibt allgemeine Gesetzmäßigkeiten für Übergänge von Systemen von einem stabilen Zustand zu einem qualitativ wesentlich verschiedenen neuen Zustand, der unter neuen Randbedingungen der stabilere ist (Theorie der Phasenübergänge). In der Physik lässt sich dies quantitativ beschreiben (Borrmann et al. 2000). Ein Phasenübergang ist umso rascher und sprunghafter, je größer das System ist. Bei finiten Systemen gibt es dagegen einen allmählichen Übergang mit einem längeren Nebeneinander von Bereichen mit alten und mit neuen Strukturen. Die Welt der wissenschaftlichen Dokumente ist zwar groß, aber endlich. Beim Übergang zu Open Access ist also Geduld gefragt.

2. Die Diskussion und Interaktion zwischen Wissenschaft und Wirtschaft ist auf dem Gebiet der Fachinformation besonders komplex und mehrschichtig. Anforderer und Empfänger von kommerziellen Diensten sind zumeist Staatsbetriebe bzw. -Bedienstete. Die Dienstleister sind nicht selbst auf dem Arbeitsgebiet der Wissenschaftler tätig.

Mit dem Aufkommen der neuen Techniken, des Mosaic-Webbrowsers und der Offenlegung der WWW-Technik durch das *CERN* 1993 wurden nicht nur binnen eines Jahres an fast allen Wissenschaftsinstituten Webserver eingerichtet. Es entwickelte sich auch eine fruchtbare offene Diskussion zwischen Experten und den Wissenschaftsverlagen. Sie führten auch zu ersten gemeinsamen Konzepten und Projektideen. So organisierte unser Institut 1995 die *Distributed Document Database (DDD)*[34], einen gemeinsamen Antrag an die Europäische Union, u. a. mit den Teilnehmern *Elsevier Science*, *Springer*, *IoPP*, *APS* und *XEROX*, der ein integriertes Konzept verteilter Dokumentenquellen bei Verlagen und Instituten an Universitäten vorsah.

Mit der fortschreitenden Erkenntnis der Qualität des Umbruchs wurde aber den Verlagen klar, dass dies die Einnahmen aus dem papierorientierten Geschäftsmodell zusammenbrechen lassen könnte und es noch keine Idee gab, ja fast keine Möglichkeit

33 Wie *UNESCO* mit dem Programm „Information für alle": http://www.unesco.de/ifap.html.
34 Vgl. http://www.physik.uni-oldenburg.de/ddd-phys/.

schien, mit OA Geld zu verdienen. Bis heute sind daher vordergründig die öffentliche, politische und die wirtschaftsöffentliche Diskussion und Argumentation von einem Festhalten der Großverlage am papierorientierten Geschäftsmodell geprägt, wie es sich auch in der Öffentlichkeitsarbeit des Börsenvereins dokumentiert.

Tatsächlich sind aber gerade die großen kommerziellen Wissenschaftsverlage mit Hochdruck dabei, sich auf die OA-Zukunft mit neuen und leistungsfähigen Produkten und Geschäftsmodellen einzustellen, wenden also einen guten Teil ihres großen Entwicklungsetats hierfür an. Ein solcher Spagat zwischen öffentlichem Festhalten an einem alt-eingeführten Produkt vergangener Technologie bei paralleler intensiver Vorbereitung eines neuen ist ein übliches, strategisches, wirtschaftliches Vorgehen für die Erhaltung der eigenen Wirtschaftskraft über die Übergangszeit hinaus. Die Öffentlichkeit und der Gesetzgeber, aber auch die Wissenschaftler selbst erhalten jedoch einen irreführenden Eindruck (Thurow 1999).

Anders als bei *Open Source* – mit seinen meist unabhängigen und nicht vernetzten Code-Erzeugern bzw. Dienstleistern – gibt es bei *Open Access* in der Wissenschaft bereits relativ homogene technische und organisatorische Strukturen, die die geforderten Aufgaben zum guten Teil wahrnehmen können: die Informationsdienstleister an den Hochschulen und wissenschaftlichen Institutionen, also die Bibliotheken, Rechenzentren und Multimediazentren. Als Geschäftsmodell läuft dies auf ein teilweises Zurückholen von seit 100 Jahren an die kommerziellen Verlage als externe Dienstleister ausgelagerten Diensten, wie z. B. technische Herstellung von Dokumenten und ihres Vertriebes, Archivierung, Verlinkung, Langzeitarchivierung etc. hinaus.

Bisher bestimmte das fachgebietsbezogene Konzept einer Zeitschrift die für ein Heft zusammengestellten Artikel. Der papiergebundene Vertriebsweg brachte das gedruckte Heft über Grossisten und Buchhändler bis zur lokalen Bibliothek, in der Hoffnung, dass ein lokaler Leser es finden und relevant finden möge. Die Bindung der einzelnen Bibliothek an den kommerziellen Verlag in Form langfristiger Paketverträge (vordefinierte Bündel von Zeitschriften müssen abgenommen werden, auch wenn es für einzelne Zeitschriften daraus keine Leser am Orte gibt) verschärft dann noch die Diskrepanz zwischen dem Angelieferten und dem Gewünschten.

Gemäß dem Terry-Report (Terry 2005) sind die Kosten für eine OA-Online-Zeitschrift sehr viel geringer: Posten auf dem Web, Download durch den Nutzer etc. sind technisch einfach und wenig kostenträchtig. Das Schreiben der Dokumente und das Referieren – wie auch das Lesen – fand ja in der wissenschaftlichen Welt schon immer in der Hochschule bzw. wissenschaftlichen Institution statt, wurde nie aus der akademischen Welt ausgelagert. Die wissenschaftlichen Bibliotheken haben zudem den großen Vorteil der Nähe zu den Erzeugern und Nutzern, außerdem erfüllen sie den staatlichen Auftrag der lokalen wissenschaftlichen Informationsversorgung.

Nun hätte man denken können, dass die kleinen und mittleren Verlage ihre große Chance hätten haben können: Sie sind flexibler, haben ein geringeres Risiko, weniger feste Kosten. Leider aber fehlt bei diesem auch technisch relevanten Umbruch in klei-

nen Firmen oft einfach die notwendige breite technische Kompetenz. Der Wagemut, diese Dinge an kompetente Spezialinstitute auszulagern, fehlt weitgehend, wird als das Geschäftsmodell ändernd, kritisch gesehen. Es gibt aber eine Fülle von neuartigen und den Prozess der Forschungsarbeit wirklich fördernden Mehrwertdiensten. Sie werden aber erst mit einer weiten Verbreitung von OA leistungsfähig. Letzteres lässt sich durch Auslagerung zu technisch kompetenten Firmen unterstützen. Einige Beispiele sind in ihrem unmittelbaren Nutzen für den Leser besonders offensichtlich: Mit *printing on demand* kann der Leser (statt eines vom Verlag auf Verdacht zusammengestellten Satzes von Artikeln) aus einer Liste von Open-Access-Artikeln sich selbst den gedruckten Band zusammenstellen. Es gibt Alerting-Dienste über alle Artikel, die vorhanden sind und die für den individuellen Leser von Interesse sein könnten. Es werden semantisch verständige Suchdienste nach individuell möglicherweise relevanten Dokumenten im Web, in Weiterverarbeitungsprogrammen etc. entwickelt.

In jüngster Zeit mehren sich die Zeichen, dass die Sprachlosigkeit zwischen Wissenschaft und Verlagen endet und die großen kommerziellen Verlage mit ihrer Finanzkraft den Einstieg in die OA-Welt vorbereiten – Beispiele sind *Elsevier Science (ES)*, *Springer* mit *Open Choice* oder *Institute of Physics Publishing (IoPP)* mit *PhysicsWeb*.

Das Jahr 2007 wird im OA-Bereich daher besonders spannend, weil einige der hier angedeuteten Entwicklungen sichtbar und erlebbar werden. Es wird eine neue, fruchtbare, professionelle Arbeitsteilung und Zusammenarbeit von Wissenschaftlern, den Universitätsdienstleistern und zum Teil auch neuen wettbewerblich kommerziellen Mehrwert-Dienstleistern geben. Zunehmend werden Hochschulen darauf dringen, dass die in ihrem Hause entstandenen wissenschaftlichen Arbeiten Open Access gelegt werden und so die Sichtbarkeit der Institution im Wettbewerb gestärkt wird.

Dabei wird die tatsächlich enge gegenseitige positive Verstärkung der Strategien bei *Open Access* von Dokumenten und *Open Source* von Programm-Codes deutlicher werden und besser genutzt werden können, in dem Maße, in dem leistungsfähigere Online-Werkzeuge entwickelt werden, die die ubiquitäre digitale Verfügbarkeit auch lokal in der Weiterverarbeitung intelligent und professionell nutzbar zu machen gestatten werden.

Literatur

Bodenschatz, E. und Smith, T. (2006), 'New Journal of Physics', in-cites. http://in-cites.com/journals/NewJofPhysics.html [13. Feb 2007].

Borrmann, P., Mülken, O. und Harting, J. (2000), 'Classification of Phase Transitions in Small Systems', *Physical Review Letters* **84**, S. 3511. http://xxx.lanl.gov/abs/cond-mat/9909184 [14. Feb 2007].

Bosc, H., Harnad, S., Hilf, E. R. und Rodrigues, E. (2006), Open Access: Threat or Blessing, *in* 'Euroscience Conference 2006', München. http://www.isn-oldenburg.de/~hilf/vortraege/esof06/ [13. Feb 2007].

Brody, T. (2006), 'Growth of OAI-PMH Institutional Archives and Contents', eprints. http://roar.eprints.org/index.php?action=analysis [13. Feb 2007].

Dobratz, S. (2006), 'Liste von Dokumenten-Servern in Deutschland'. http://www.dini.de/dini/wisspub/dokuserver.php [14. Feb 2007].

Harnad, S., Brody, T., Vallières, F., Carr, L., Hitchcock, S., Yves, G., Charles, O., Stamerjohanns, H. und Hilf, E. R. (2004), 'The Access/Impact Problem and the Green and Gold Roads to Open Access', *Serials Review* **30**(4), S. 310–314. http://eprints.ecs.soton.ac.uk/10209/ [14. Feb 2007].

Harnad, S., Hilf, E. R., Laloé, F., Sale, A. und Swan, A. (2007), 'Personal, Institutional and Central Repositories: Growth and Incentives'. Noch nicht veröffentlicht.

Hilf, E. R. und Severiens, T. (2006), 'Digitale Verlagsdienste in einer Open Access Welt: Zusammenstellung möglicher Add-On Dienste'. http://www.isn-oldenburg.de/~hilf/kampffmeyer4.pdf [13. Feb 2007].

Korb, N. (2006), 'Sammlung, Verzeichnung und Archivierung durch die Deutsche Nationalbibliothek – Angaben zur Statistik', Deutsche Nationalbibliothek. http://info-deposit.d-nb.de/ablieferung/online_hss/hss_statis.htm [13. Feb 2007].

Kuhlen, R., Beger, G. und Degkwitz, A. (2004), 'Göttinger Erklärung zum Urheberrecht für Bildung und Wissenschaft', Aktionsbündnis „Urheberrecht für Bildung und Wissenschaft". http://www.urheberrechtsbuendnis.de/GE-Urheberrecht-BuW-Mitgl.pdf [29. Jan 2007].

Neumann, J. (2006), Auf dem Weg zu einem Open-Access-Geschäftsmodell – Erfahrungsbericht German Medical Science, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 319–336. http://www.opensourcejahrbuch.de/ [12. Dez 2006].

Sahu, D. K., Gogtay, N. J. und Bavdekar, S. B. (2005), Effect of Open Access on Citation Rates for a Small Biomedical Journal, *in* 'Fifth International Congress on Peer Review and Biomedical Publication, Chicago', Medknow Publications, Mumbai, Indien. http://openmed.nic.in/1174/ [13. Feb 2007].

Severiens, T. und Hilf, E. R. (2006), Zur Entwicklung eines Beschreibungsprofils für eine nationale Langzeit-Archivierungs-Strategie – ein Beitrag aus der Sicht der Wissenschaften, Studie, nestor-Materialien 7, nestor Kompetenznetzwerk Langzeitarchivierung. http://nbn-resolving.de/urn:nbn:de:0008-20051114021 [13. Feb 2007].

Swan, A. und Brown, S. (2005), 'JISC reports on Open Access Usage and Expectations'. http://www.ecs.soton.ac.uk/~harnad/Temp/Romeo/romeosum.html [13. Feb 2007].

Terry, R. (2005), Publication Costs are Research Costs, *in* 'Conference on Open Access to Knowledge in the Sciences and Humanities', CERN. http://www.zim.mpg.de/openaccess-cern/presentation-terry.ppt [13. Feb 2007].

Thurow, L. C. (1999), *Building Wealth: The new Rules for Individuals, Companies, and Nations in a Knowledge-Based Economy*, Collins, New York.

Open Source, Open Content und Open Access – Gemeinsamkeiten und Unterschiede

RETO MANTZ

(CC-Lizenz siehe Seite 563)

Open Source, Open Content und Open Access werden häufig und berechtigterweise im gleichen Kontext genannt und als gemeinsame oder zumindest aufeinander basierende Phänomene betrachtet. Mit einem genauen Blick auf die jeweilige Motivation und die rechtlichen Grundlagen lassen sich allerdings einige Unterschiede ausmachen, die sich auch im praktischen Einsatz auswirken können. Anhand der Gemeinsamkeiten, aber auch der Unterschiede, sollen die verschiedenen Konzepte erläutert und in Beziehung zueinander gesetzt sowie offene Fragen und Konfliktpotenziale aufgezeigt werden.

Schlüsselwörter: Open Access · Open Content · offenes Publizieren

1 Einleitung

Die Veröffentlichung von Werken unter Lizenzen, die eine weitgehend freie und vor allem kostenlose Verwendung des Werks erlauben, nimmt stetig zu. Während bereits Ende der 1980er Jahre Software auf diese Weise verbreitet wurde, hat sich in den letzten Jahren die freie Veröffentlichung von anderen Inhalten und auch wissenschaftlichen Werken als mindestens paralleler Verbreitungsweg etabliert. Für die unterschiedlichen Formen haben sich schlagwortartige Bezeichnungen eingebürgert: Während die freie Veröffentlichung und Verbreitung von Software unter *Open Source* firmiert, fallen unter *Open Content* Inhalte jeglicher Art. Hinzu kommt der Begriff des *Open Access*, der eher den freien Zugang zu wissenschaftlichen Werken öffentlich finanzierter Forschung erfasst.

In diesem Beitrag sollen die Gemeinsamkeiten, aber auch die Unterschiede im Hinblick auf die rechtliche Gestaltung der drei Konzepte beleuchtet werden. Dabei wird ein besonderes Augenmerk auf die Grundlagen der Konzepte, also insbesondere die Motivation der Beteiligten bei Freigabe ihrer Werke sowie die eingesetzten rechtlichen

Methoden, gelegt. Zudem soll dieser Beitrag offene rechtliche Probleme oder mögliche Problemsituationen aufzeigen, die teilweise gerade in der Unterschiedlichkeit der Konzepte begründet sind.

Unbestritten ist, dass der rechtliche Anknüpfungspunkt bei der Lizenzierung im Urheberrecht zu finden ist: Die Verbreitung der Werke wird über urheberrechtliche Nutzungsrechte geschützt und gefördert. Diese Konstruktion ist jedoch lediglich Mittel zum Zweck. Die Motivation für die Verwendung offener Modelle wird meist auf andere individuelle Rechte bzw. Grundrechte zurückgeführt. Die einschlägigen Rechte sollen für die unterschiedlichen Konzepte jeweils kurz bestimmt werden.

Die Rechtsübertragung erfolgt aufgrund von Lizenzverträgen. Bei aller Ähnlichkeit finden dennoch unterschiedliche Lizenzverträge für die unterschiedlichen Formen Verwendung. Diese Differenzen sollen ebenfalls behandelt und – soweit möglich – begründet werden.

2 Open Source

Open Source ist der Prototyp der Debatte um offene Ressourcen. Es ist bereits intensiv Objekt wissenschaftlicher Erörterungen gewesen und die lizenzrechtlichen Schlussfolgerungen zu diesem Bereich hatten und haben maßgeblichen Einfluss auf die Entwicklung der Lizenzverträge von Open Content und Open Access.

Die Open-Source- bzw. Free-Software-Bewegung ist – sehr kurz gefasst – entstanden, um eine Alternative zu proprietärer Software, speziell Betriebssystemen wie *UNIX*, zu bieten. Daraus hat sich eine Bewegung entwickelt, deren Ziel die Schaffung, Verbesserung und freie Verbreitung von Computerprogrammen ist.[1] Rechtlich ist sie stark im Gedanken der Redefreiheit des Art. 5 Abs. 1 GG verankert.[2]

Die Entwicklung von Open-Source-Programmen basiert wesentlich auf einem ausgeprägten Gemeinschaftsgedanken. Häufig beginnt eine Einzelperson oder ein kleines Team mit der Entwicklung eines Produkts und stellt dieses anderen auf Basis einer Open-Source-Lizenz wie der *GNU General Public License (GPL)* zur Verfügung.[3] Die *GPL* ist nur der wichtigste Vertreter der Open-Source-Lizenzen. Neben diese treten andere, teilweise offenere, teilweise restriktivere Lizenzen. Zu den Bekannteren zählen z. B. die *LGPL*, die *Mozilla Public License* und die *BSD Software License*.[4] Bei Interesse bilden sich rund um das Projekt größere Gruppen. Insbesondere bei komplexen Programmen, deren Entwicklungsaufwand nur in Mannmonaten oder Mannjahren zu messen ist, ist eine große Gemeinschaft zur Bearbeitung und Entwicklung der Programme notwendig. Diese Gemeinschaften zeichnen sich erstens regelmäßig

1 Zur Entwicklung von Open Source siehe Rasch (2000).

2 „Free Software—Free as in freedom [free speech][. . .]" (Grassmuck 2004, S. 285).

3 Zur Community-Bildung siehe Stürmer und Myrach (2006), vgl. auch Raymond (1999).

4 Eine umfassende Übersicht über die zur Verfügung stehenden Lizenzen findet sich unter http://www.ifross.de/ifross_html/lizenzcenter.html.

durch eine gemeinsame Verbundenheit in der Motivation aus.[5] Ein weiteres wichtiges Merkmal ist, dass die Mitarbeit an solchen Projekten absolut freiwillig erfolgt. Zwar erzwingt die von der GNU-Lizenz vorgegebene Copyleft-Konstruktion,[6] dass bearbeitete Werke bei Verbreitung immer auch im Quelltext veröffentlicht werden müssen, aber die Entscheidung darüber, ob und wie man ein Programm weiterentwickelt, obliegt dem Programmierer selbst. Open-Source-Gemeinschaften können jedenfalls in kürzester Zeit wachsen und wieder schrumpfen und eine hohe Personalfluktuation aufweisen. Dem steht nicht entgegen, dass mittlerweile auch von Unternehmen bezahlte Programmierer an Open-Source-Projekten arbeiten. Selbstverständlich verfolgen die Unternehmen und somit auch die Mitarbeiter dabei kommerzielle Interessen und sind in der Lage, die Entwicklung von Produkten maßgeblich zu beeinflussen. Dennoch überwiegt die Anzahl der „freien" Entwickler: Der Gemeinschaftsgedanke bleibt maßgebliches Merkmal in der Entwicklung von Open-Source-Projekten. Vielmehr ist davon auszugehen, dass die Unternehmen diesen fördern und nutzen, um auch unternehmensfremde Programmierer einzubinden.

Auch die rechtliche Konstruktion bei der Verbreitung von Open-Source-Werken hat Niederschlag in vielen wissenschaftlichen Publikationen gefunden und ist in weiten Bereichen als aufgearbeitet anzusehen.[7]

Die *GPL* stellt demnach eine Vertragsvorlage dar, durch die der Urheberrechtsinhaber, also der Ersteller eines Werkes, dem zukünftigen Nutzer das Angebot auf Abschluss dieses Vertrages macht. Das entsprechende Vertragsangebot wird in der Verbindung zwischen Werk und Vertragstext „gespeichert".[8] Der Nutzer, von dem der Urheber in aller Regel nichts weiß, kann es konkludent annehmen, wobei nach § 151 Satz 1 2. Alt. BGB auf den Zugang der Annahme verzichtet wird. Mit Abschluss des Lizenzvertrages räumt dann der Urheber dem Nutzer nach § 31 Abs. 2 UrhG weitgehende einfache Nutzungsrechte am Werk entsprechend den Vertragsbedingungen ein. Die Absicherung der Bedingungen, allen voran das Copyleft-Prinzip, erfolgt dadurch, dass die entsprechenden Willenserklärungen unter eine auflösende Bedingung nach § 158 Abs. 1 BGB gestellt werden. Mit Eintritt einer Verletzung der Vertragsbedingungen fallen die Nutzungsrechte an den Urheber zurück. Alle weiteren nicht durch urheberrechtliche Schranken gedeckten Handlungen am Werk oder den Bearbeitungen, wie z. B. die weitere Verbreitung, sind anschließend rechtswidrig und können gerichtlich untersagt werden. Die rechtliche Wirksamkeit der *GPL* und damit auch anderer Open-Source-Lizenzen in Deutschland per se wird im Grunde seit dem wegweisenden Urteil des Landgerichts München I nicht mehr bezweifelt.[9] Das Landgericht hatte im Jahr 2004 die GPL als wirksame Vertragsvorlage eingestuft und auch die Wirksamkeit des Rückfalls der Rechte bei einer Verletzung bestätigt.

5 Zur Motivation ausführlich in Merten und Meretz (2005, S. 294).
6 Die BSD-Lizenzen enthalten z. B. keine solche Klausel.
7 Siehe Jaeger und Metzger (2006) oder Spindler (2004).
8 Allgemein zur „Speicherung" von Willenserklärungen in Dokumenten bei John (1984, S. 391 ff.).
9 Speziell im Urteil des LG München I MMR 2004, 693.

Im zu entscheidenden Fall hatte ein Hersteller ein auf einem GPL-geschützten Werk basierendes Produkt erstellt und es ohne den geforderten Hinweis sowie ohne Offenlegung des Quelltextes vertrieben. Die Klage eines Miturhebers des Werks vor dem Landgericht hatte Erfolg. Lediglich die Einbeziehung sowie die Wirksamkeit einiger Klauseln, insbesondere der Haftungsklauseln, sind nach den Regeln über allgemeine Geschäftsbedingungen entsprechend §§ 305 ff. BGB bedenklich. Kürzlich hat das Landgericht Frankfurt die Gültigkeit der *GPL* erneut bestätigt.[10] Dabei hatte der Hersteller eines funknetzwerkfähigen Geräts GPL-geschützten Programmcode aus dem Linux-Kernel verwendet, dem Produkt aber weder die *GPL* beigelegt noch den Quelltext offen gelegt. Nach richtiger Auffassung des Landgerichts Frankfurt handelte er diesbezüglich lizenzwidrig.

Problematisch bleibt, dass am Werk eine Vielzahl an Urhebern mitwirkt und deren Beitrag im Einzelnen häufig nicht feststellbar bzw. eingrenzbar ist. Dadurch stellen sich vor allem in der Feststellung der Anspruchsberechtigten und der Mitglieder der Urhebergemeinschaften Probleme,[11] was sich nicht zuletzt daran zeigt, dass im der Entscheidung des Landgerichts Frankfurt zugrunde liegenden Fall der Kläger sich von den Ersturhebern ausschließliche Nutzungsrechte erteilen ließ, um überhaupt seine Klageberechtigung darlegen zu können.[12] Lassen sich die ursprünglichen Urheber nicht mehr oder nicht eindeutig ermitteln, können bereits hier Schwierigkeiten auftreten.

Hilfreich auch für die spätere Betrachtung ist die Benennung der Akteure. Bei Open Source sind die Programmierer, also die Urheber, auf der einen, die Nutzer, die selbstverständlich auch selbst zu Urhebern werden können, auf der anderen Seite beteiligt. Die Hersteller von kommerziellen Konkurrenzprodukten sind eher als Gegenspieler anzusehen. An der Open-Source-Produktionskette nehmen sie meist nicht teil. Eine Ausnahme hierzu bilden diejenigen Hersteller, die sich selbst zu Open Source bekennen, das Modell unterstützen und es sich nutzbar machen.[13]

3 Open Content

Open Content ist das Schlagwort für eine Bewegung, die von den Erfahrungen bei der Verbreitung von Open-Source-Software ausgehend ganz allgemein die Zirkulation

10 Siehe dazu LG Frankfurt, Urteil vom 6. 9. 2006 – 2-6 0 224/06 unter http://www.jbb.de/urteil_lg_frankfurt_gpl.pdf, dazu auch Martens (2006) und Diedrich (2006).

11 So wird teilweise diskutiert, ob die Beteiligten eine Gesellschaft nach §§ 705 ff. BGB bilden und welche Folgen dies hätte. Die h. M. lehnt diese Konstruktion aber für den Regelfall der Open-Source-Erstellung ab, siehe Heussen (2000, S. 332), Jaeger und Metzger (2006, S. 197 ff.), Grützmacher (2006, S. 109), Koch (2000, S. 277 f.), Sujecki (2005, Abs. 8), Sester (2000, S. 801), Jaeger und Schulz (2005, S. 85) und Spindler (2005, S. 542).

12 Siehe zu diesem Fall LG Frankfurt, Urteil vom 6. 9. 2006 – 2-6 0 224/06 unter http://www.jbb.de/urteil_lg_frankfurt_gpl.pdf.

13 So z. B. *Red Hat*, dazu Knoblich (2006) und zu *Novell*: Smid (2006).

von Inhalten umfasst. Die dahinterstehende Idee ist, dass Informationen als nicht–körperliche und leicht jederzeit vervielfältigbare Güter nach Möglichkeit jedem und jederzeit zur Verfügung stehen sollten. Ein eindrucksvolles Beispiel hierfür bieten Wikis im Allgemeinen und *Wikipedia* im Speziellen.[14]

Neben Open-Source-Programmen als Werke treten dadurch ganz allgemein Texte, Bilder, Tonsequenzen, Filme, Prozessorarchitekturen usw. auf. Der Fantasie sind insoweit keine Grenzen gesetzt. Auch bei Open Content gibt es eine große Gemeinschaft der Beitragenden. An *Wikipedia* z. B. arbeiten viele Freiwillige, wobei ebenfalls anonymisierte bzw. pseudonymisierte Beiträge möglich sind, so dass sich die Identität aller Urheber faktisch nicht feststellen lässt. Häufiger als bei Open Source kommt es jedoch auch vor, dass nur einzelne Personen an einem Werk arbeiten und sich um dieses keine Bearbeitungsgemeinschaften bilden. Open Content hat sich in vielerlei Hinsicht der positiven Erfahrungen der Open-Source-Bewegung bedient. Insbesondere werden Lizenzverträge benutzt, die den Open-Source-Lizenzen sehr ähnlich sind. Dennoch bestehen bereits in der Lizenzvertragsgestaltung deutliche Unterschiede zu Open Source. Open-Content-Lizenzen sind häufig modular aufgebaut. Bekanntestes Beispiel hierfür sind die Creative-Commons-Lizenzen.[15] Modularität bedeutet, dass der Urheber aus einer Vielzahl von Vertragsmodalitäten die für ihn passende Variante wählen kann. Die Heterogenität der Lizenzverträge wird demnach bei Open Content bereits über die Modularität erreicht, wohingegen bei Open Source mehrere monolithische Lizenzvorlagen wie die *GPL* oder die *Lesser GPL (LGPL)* nebeneinander stehen.[16] Im Ergebnis hat natürlich der Urheber sowohl bei Open Source als auch bei Open Content die Wahl; Open Content ist aber bereits auf diese Differenzierungen ausgelegt. Ein Modul, das bei Open Source zudem praktisch unbekannt ist, ist das Verbot von Bearbeitungen – bei den Creative-Commons-Lizenzen „No Derivatives" (ND) genannt. Während Open Source ganz wesentlich darauf beruht, dass Entwicklergemeinschaften ein Produkt weiterentwickeln, ist bei Open Content also die alleinige Bearbeitungsbefugnis des Urhebers durchaus üblich. Die Folge dieser Entscheidung ist jeweils, dass zwar die Information der Allgemeinheit zur Verfügung steht, aber nur über die urheberrechtlichen Schranken, wie z. B. die Zitierfreiheit nach § 51 UrhG, in den Diskurs einbezogen werden kann. Weiterentwicklungen bleiben dem Urheber vorbehalten. Die Problematik der Urhebergemeinschaften und ihrer rechtlichen Behandlung stellt sich in diesem speziellen Fall nicht. Ganz allgemein tritt die Open-Content-Gemeinde aber sehr stark für eine möglichst schrankenfreie Verbreitung der Inhalte ein. Kommerzielle Nutzung sowie Bearbeitungen sollten soweit wie möglich erlaubt werden.[17]

Auch zeigt sich, dass die Problempunkte bei Open Content weitgehend andere als

14 Siehe dazu im Open Source Jahrbuch 2005 Danowski und Voß (2005).

15 Auch die Creative-Commons-Lizenzen haben mittlerweile gerichtliche Anerkennung gefunden, siehe Mantz (2006*a*).

16 Mit dem Versionssprung auf *GPL v3* könnte sich dies allerdings ändern, siehe Jaeger (2006).

17 Vgl. z. B. Waidele (2006), ebenso Möller (2006).

bei Open Source sind. So dreht sich ein Großteil der Diskussionen um die Möglichkeit des Urhebers, die kommerzielle Nutzung zu untersagen.[18] Schwierigkeiten bereitet hierbei speziell die Frage, wann eine Nutzung als kommerziell einzustufen ist.[19] Daraus können sich erhebliche rechtliche Unsicherheiten ergeben. Bei Open Content tritt neben die Bearbeitung als Mittel der „Weiterentwicklung" vor allem auch die Kompilation in Sammelwerken,[20] die durch das Verbot der kommerziellen Nutzung an Brisanz gewinnt. Dies zeigt sich nicht zuletzt daran, dass die Lizenzen spezielle, teilweise komplizierte Regelungen hinsichtlich der Verwendung von Sammelwerken aufweisen.[21] Grund hierfür ist sicher ebenfalls, dass unter Open Content auch Werke fallen, deren „Weiterentwicklung" sich erheblich schwieriger gestaltet als bei Computerprogrammen. Auch zeigen diese Module, dass seitens der Urheber durchaus Bedarf daran besteht, die Nutzung nicht vollständig freizugeben bzw. über den reinen Copyleft-Gedanken, der in der Terminologie von *Creative Commons* „Share Alike" genannt wird, hinaus einzuschränken. Rechtlich knüpft Open Content ebenfalls an die Kommunikationsfreiheiten im weiteren Sinne nach Art. 5 Abs. 1 GG an.

Als relevante Akteure treten erneut Urheber und Nutzer auf. Teilweise besteht auch eine echte Konkurrenzsituation, z. B. zwischen der *Wikipedia* und kommerziellen Lexikonherstellern. Diese Konkurrenz ist aber nicht so stark ausgeprägt und in den Prozess einbezogen wie bei Open Source. Neue Akteure hingegen sind die Verwertungsgesellschaften. Diese vertreten z. B. bei Werken der Musik die Urheber und nehmen deren Rechte gegenüber Dritten wahr. Daraus ergibt sich ein gänzlich neues Spannungsfeld. Werke, die bereits von einem Wahrnehmungsvertrag des Urhebers mit einer Verwertungsgesellschaft erfasst werden, kann der Urheber nicht unter eine Open-Content-Lizenz stellen. Da vorher der Verwertungsgesellschaft ausschließliche Nutzungsrechte erteilt wurden, ist dem Urheber eine Einräumung einfacher Nutzungsrechte durch die Verbreitung unter einer Open-Content-Lizenz rechtlich nicht mehr möglich.[22] Besonders für den Nutzer ist die Situation denkbar ungünstig. Da es keinen gutgläubigen Erwerb von Nutzungsrechten gibt,[23] steht das Werk zwar augenscheinlich unter einer Open-Content-Lizenz, tatsächlich ist die Nutzung aber nicht so frei wie es der Lizenzvertrag verspricht. Entsprechende Gerichtsverfahren sind hierzulande noch nicht bekannt geworden. In Spanien hatte ein Gericht den Fall zu

18 Eingehend dazu Möller (2006).

19 Der Entwurf einer Richtlinie zur Entscheidung über den Tatbestand der Kommerzialität ist erhältlich unter http://wiki.creativecommons.org/DiscussionDraftNonCommercial_Guidelines.

20 Dazu auch Liang (2004).

21 Zum Beispiel Ziffer 4 lit. a und Ziffer 7 lit. a Satz 2 der *Creative Commons License*, siehe unter http://creativecommons.org. Außerdem § 2 Abs. 3 der *Digital Peer Publishing License*, siehe unter http://www.dipp.nrw.de/lizenzen/dppl.

22 Die *Creative Commons* weist auch darauf hin, dass GEMA-Mitglieder Creative-Commons-Lizenzen nicht einsetzen können, siehe http://de.creativecommons.org/faq.html, zur Problematik auch Herrmann und Kösch (2004).

23 Siehe dazu BGHZ 5, 116 (119); BGH GRUR 1959, 200 (203); siehe auch Wandtke und Grunert (2006), Schricker (2006) und Hertin (1998, Rn. 9).

entscheiden, dass die berechtigte Verwertungsgesellschaft von einem Barinhaber, der in seiner Bar nur unter CC-Lizenz stehende Musik spielte, entsprechende Gebühren verlangte. Dem Gericht zufolge bestand aber aufgrund der vorhandenen CC-Lizenzen keine Pflicht seitens des Beklagten, Gebühren an die Verwertungsgesellschaft abzuführen. Die Urheber hätten aus dem „moralischen und wirtschaftlichen Recht" an ihrem Werk die Berechtigung, ihre Werke zu verbreiten, ohne Ansprüche der Verwertungsgesellschaft den Nutzern gegenüber zu begründen.[24] Ob ein deutsches Gericht ebenso entscheiden würde ist fraglich. Mit zunehmendem falschen Einsatz von Open-Content-Lizenzen, z. B. durch Musiker, ist es aber nur eine Frage der Zeit, bis ein solches Gerichtsverfahren betrieben wird.

Für viel Spannung dürfte auch in Zukunft die weitere Entwicklung der entsprechenden Lizenzen sorgen. Grund hierfür ist, dass die für Open Source relevante *GPL* weiterentwickelt und vermutlich im ersten Quartal 2007 den Versionswechsel auf Version 3.0 vornehmen wird.[25] Gleichzeitig wird mit Hochdruck an der Modernisierung der Creative-Commons-Lizenzen gearbeitet.[26] Beachtenswert ist, dass die Diskussionspunkte sich teilweise decken. So sollen beide Vertragswerke zukünftig Regelungen hinsichtlich Digital-Rights-Management-Systemen (DRM) enthalten.[27] Zudem öffnen sich die Creative-Commons-Lizenzen immer mehr auch in Richtung von Programmen bzw. Programmtexten, wie sich ebenfalls an der Debatte um DRM zeigt. Creative-Commons-Lizenzen sollen demnach tatsächlich für alle möglichen Inhalte geeignet sein und treten damit in eine Konkurrenzsituation zu den klassischen Open-Source-Lizenzen. Diese könnte durch Kompatibilitätsklauseln zumindest entschärft werden. Streitpunkt ist nämlich auch die Vereinbarkeit von verschiedenen Lizenzen. Dabei dreht sich die Frage jeweils hauptsächlich darum, ob Inhalte, die unter einer bestimmten Lizenz stehen, mit Inhalten unter anderen Lizenzen vermengt werden können und welche Lizenz anschließend bei Derivaten zu wählen ist. So könnten möglicherweise Bildwerke unter einer Creative-Commons-Lizenz mit Quelltexten eines Programms und der *GPL* vereint werden, wobei das gemeinsame Dach entweder beide Lizenzen oder sogar nur die Creative-Commons-Lizenz sein könnten. Dadurch könnten die Unterschiede zwischen Open Source und Open Content zumindest lizenzrechtlich weiter schwinden.

24 Siehe dazu im Spanischen Urteil *Juzgado de Primera número seis de Badajoz*, Urteil vom 17. 2. 2006 – Verfahrensnr. 761/2005, Urt. Nr. 15/2006, dazu auch http://creativecommons.org/weblog/entry/5830.

25 Ausführlich dazu http://gplv3.fsf.org, siehe auch Moglen (2006), Jaeger (2006), Jaeger und Metzger (2006) oder auch Mantz (2006*b*).

26 Aktuelle Entwürfe und Diskussionen unter http://wiki.creativecommons.org/License_versions#3.0.

27 Die Diskussion bei *Creative Commons* lässt sich gut in den Mailing-Listen-Archiven verfolgen: http://lists.ibiblio.org/pipermail/cc-licenses/2006-August/003855.

4 Open Access

Open Access steht inhaltlich und teilweise auch rechtlich Open Content sehr nahe. Die Spezialität liegt darin, dass es sich um die Verbreitung von wissenschaftlichen Werken, meist also Textwerken und Begleitmaterial, handelt.

Die Open-Access-Bewegung gründet sich zu großen Teilen auf der sogenannten Publikationskrise: Mit einer steigenden Anzahl von Fachzeitschriften, steigenden Subskriptionspreisen und demgegenüber sinkenden öffentlichen Mitteln sehen sich die wissenschaftlichen Institutionen, allen voran die Bibliotheken, immer weniger in der Lage, die in wissenschaftlichen Werken gesammelten wissenschaftlichen Erkenntnisse einzukaufen und vorzuhalten. Verschärfend wirkt der Umstand, dass durch die hohen Preise im Grunde mit öffentlichen Mitteln nur diejenigen wissenschaftlichen Ergebnisse erworben werden, die vorher in der Forschung zumeist durch öffentliche Förderung gewonnen wurden. Es liegt somit eine Doppelfinanzierung vor, die als zunehmend unerträglich empfunden wird.[28]

Als Reaktion entstand die *Budapest Open Access Initiative* sowie in der Folge die *Berliner Erklärung über offenen Zugang zu wissenschaftlichem Wissen*[29]. Nach der *Berliner Erklärung* soll der Zugang zu Wissen offen sein. Dies umfasst die Rechtseinräumung zur Nutzung, Vervielfältigung und Verbreitung[30] – auch von Bearbeitungen – unter Beibehaltung der korrekten Namensnennung, wobei nicht auf spezielle Lizenzen verwiesen wird. Die Erstellung von körperlichen Vervielfältigungsstücken soll zumindest in geringem Umfang zum persönlichen Gebrauch erlaubt werden. Die Werke sollen des Weiteren dauerhaft elektronisch vorgehalten werden. Dafür werden sogenannte Archivserver bzw. *repositories*[31] verwendet. Während Open Source und Open Content diese Dauerhaftigkeit nicht explizit verfolgen, ist sie bei Open Access also eines der Primärziele, um einmal gewonnenes Wissen zu erhalten.

Als Resultat sollten nach der *Berliner Erklärung* alle Wissenschaftler ihre Werke in solchen *repositories* hinterlegen. Für viele Wissenschaftler gibt es allerdings wenig Alternativen zur Veröffentlichung über große und anerkannte Verlage. Der Ort der Veröffentlichung bestimmt maßgeblich die wissenschaftliche Aufmerksamkeit (sog. *impact*) und damit auch den Reputationsgewinn des Wissenschaftlers.[32] Diese Be-

28 Siehe dazu Pflüger und Ertmann (2004, S. 437), Hilty (2006, S. 181 ff.), Neumann (2006, S. 321 f.) und Sietmann (2006, S. 192).

29 Die originale Erklärung hier: http://www.zim.mpg.de/openaccess-berlin/berlindeclaration.html. Die Übersetzung unterscheidet sich in wesentlichen Punkten vom Original: http://www.mpg.de/pdf/openaccess/BerlinDeclaration_dt.pdf.

30 Ob das Recht der öffentlichen Zugänglichmachung erfasst sein soll, lässt sich dem Text nicht explizit entnehmen.

31 Ein Beispiel für ein *repository* unter http://www.arxiv.org.

32 Siehe dazu Pflüger und Ertmann (2004, S. 436 f.), inwieweit der *impact* durch die freie Veröffentlichung sogar erhöht wird, ist ebenfalls Diskussionsgegenstand, dazu mehr unter http://www.ma.uni-heidelberg.de/bibl/a-z/open_access/impact.html und in der DFG-Studie (Over et al. 2005). Zur Berechnung des *impact factor* siehe Dong et al. (2005).

standsaufnahme gilt aber bereits jetzt nicht mehr für alle Disziplinen. Insbesondere in den Naturwissenschaften existieren bereits heute anerkannte und damit auch bei den Urhebern akzeptierte Open-Access-Zeitschriften. Erklärtes Ziel vieler Universitäten ist der Ausbau des Angebots an offen zugänglichen Informationsquellen, in die Wissenschaftler ihre Werke einstellen.[33] Eben für den Bereich wissenschaftlicher Publikationen wurde mit der *Digital Peer Publishing License*[34] (*DPPL*) vom Ministerium für Wissenschaft und Forschung des Landes Nordrhein-Westfalen ein speziell auf Open-Access-Zeitschriften bzw. -Werke zugeschnittener Lizenzvertrag entwickelt.[35] Die *DPPL* ist ebenso wie die Creative-Commons-Lizenzen modular aufgebaut. Insbesondere kann der Urheber auch hier die Bearbeitung verbieten.[36] Bei wissenschaftlichen Werken wird davon häufig Gebrauch gemacht, da der Urheber für die unter seinem Namen erscheinenden Beiträge in besonderem Maße einstehen muss. Dem Ziel, die Zugänglichkeit des Werkes zu gewährleisten und damit den wissenschaftlichen Diskurs und die wissenschaftliche Weiterentwicklung zu fördern, genügt der Urheber auch mit dieser Form der Verbreitung, allerdings unter Widerspruch zur *Berliner Erklärung*. Damit wird eines der Probleme der *Berliner Erklärung* umgangen. In ihr enthalten ist die Erlaubnis zur Bearbeitung und anschließenden Veröffentlichung unter dem Namen des Ursprungsurhebers. Bearbeitungen sollten aber vielmehr eindeutig als solche gekennzeichnet werden. In der vorliegenden Form schützt die *Berliner Erklärung* folglich nicht vor Verfälschungen, die dann unter dem Namen des Wissenschaftlers verbreitet werden könnten. Ihm verbleibt nur der Entstellungsschutz nach § 14 UrhG.

Interessant ist zudem, dass die *DPPL* entsprechend § 3 Abs. 2 und § 4 *DPPL* nicht jede Form der Entgeltsicherung verbietet. Bietet z. B. die zentrale Archivierung einen Mehrwert, so darf zwar nicht für das Werk an sich, durchaus aber für den Zugang ein Entgelt verlangt werden. Der Zugang kann auch durch technische Schutzmaßnahmen, beispielsweise eine Nutzerkennung, gesichert sein.[37] Im Hinterkopf sollte behalten werden, dass der Nutzer nach einmaligem Zugang selbstverständlich alle im Lizenzvertrag festgehaltenen Rechte, also u. a. das Recht der öffentlichen Zugänglichmachung, erwirbt.

Darin zeigt sich bereits die im Vergleich zu Open Content unterschiedliche Motivation. Die wissenschaftliche Diskussion und Weiterentwicklung auf Basis der Wissenschaftsfreiheit nach Art. 5 Abs. 3 GG stehen im Vordergrund. Für diese ist natürlich die freie Verfügbarkeit notwendig, sie wird jedoch nicht primär auf die Redefreiheit

33 So sollen z. B. Ergebnisse aus DFG-geförderten Studien in Zukunft unter Open Access zur Verfügung stehen, dazu mehr unter http://www.dfg.de/lis/openaccess.

34 http://www.dipp.nrw.de

35 Die *DPPL* soll die konsequente Umsetzung des Open-Access-Ansatzes ermöglichen, vgl. http://www.dipp.nrw.de/lizenzen/dppl, sie könnte aber auch als Open-Content-Lizenz bezeichnet werden.

36 Bei der Basisversion der *DPPL* ist dies Standard. Der Urheber kann auch die Bearbeitung nur von Teilen des Werks erlauben (modulare *DPPL*), gedacht wurde dabei z. B. daran, den Text zu schützen, aber die als Beleg dienenden Formeln zur Bearbeitung und Weiterentwicklung freizugeben, siehe http://www.dipp.nrw.de/lizenzen/faq/veraenderung/faq23.

37 Zu diesem Themenbereich Knauf (2006, S. 105 f.).

nach Art. 5 Abs. 1 GG gestützt. Der Gemeinschaftsgedanke im Sinne einer gemeinschaftlichen Bearbeitung ist bei Open Access in der Regel weniger stark ausgeprägt und bezieht sich im Wesentlichen auf die wissenschaftliche Fortentwicklung. Dadurch tritt ein rechtliches Problem von Open Source und Open Content häufig gar nicht auf: Der bzw. die Urheber sind eindeutig benennbar. Die Bildung von unübersichtlichen Gesellschaften bzw. Urhebergemeinschaften entfällt weitgehend.

Weitere Akteure, die bei Open Source keine, bei Open Content nur eine geringe, bei Open Access hingegen eine wesentliche Rolle spielen, sind die Verlage. Während bei Open Content zwar durchaus Verlage als Konkurrenten auftreten, sind sie im Bereich der Wissenschaftspublikation diejenigen, die den Zugang zu den etablierten und anerkannten und damit reputationsträchtigsten Publikationswegen kontrollieren. Im Bereich des Open Access wird deshalb auch deutlich zwischen verschiedenen Arten der Open-Access-Publikation unterschieden. Die „Golden Road" ist hierbei diejenige, bei der der Autor das Werk tatsächlich und originär unter einer offenen Lizenz der Allgemeinheit zur Verfügung stellt. Die „Green Road" hingegen beschreibt die erstmalige Veröffentlichung des Werks über einen Verlag als Mittler und anschließend bzw. später die freie Veröffentlichung, z. B. auf der eigenen Homepage.

Meist räumt der Urheber den Verlagen die ausschließlichen Nutzungsrechte ein. Diese Konstellation ist bei Open Source gar nicht, bei Open Content nur ausnahmsweise bekannt. Neben die Rechtsbeziehung zwischen Urheber und Nutzer tritt folglich diejenige zwischen Urheber und Verlag, woraus sich einschneidende Auswirkungen ergeben können. Will der Wissenschaftler das Werk trotz Einigung mit dem Verlag unter einer Open-Access-Lizenz veröffentlichen, so muss er die vertraglichen Grundlagen mit dem Verlag entsprechend gestalten. Eine Beschränkung auf die ausschließliche Einräumung von Nutzungsrechten nur für die kommerzielle Nutzung, wie sie z. B. das *SPARC Author's Addendum*[38] vorsieht, ist beispielsweise nicht möglich, da die Aufteilung der Urheberrechte auf die verschiedenen Nutzungsarten nur realisierbar ist, wenn es sich um wirtschaftlich-technisch abgrenzbare Nutzungen handelt.[39] Die Unterscheidung zwischen kommerzieller und nicht-kommerzieller Nutzung ist aber lediglich wirtschaftlicher Natur und damit nicht als Abgrenzungskriterium tauglich.

Selbst die originäre Veröffentlichung als Open Access mit anschließendem Abschluss eines Verlagsvertrages birgt Haftungsrisiken für den Urheber und Rechtsunsicherheit bei den Nutzern. Grund dafür ist, dass die Belastung des Urheberrechts nach § 33 UrhG nur diejenigen Nutzer schützt, die den Lizenzvertrag mit dem Urheber abgeschlossen haben, bevor er die folgende ausschließliche Rechtseinräumung an den Verlag vorgenommen hat.[40] Eine analoge Anwendung des § 33 UrhG zugunsten der Nutzer ist nicht möglich, mindestens ist die Frage aber als ungeklärt anzusehen. Die Verwendung der Open-Access-Lizenzen beschränkt somit nicht bereits das dem

38 http://www.arl.org/sparc/author/addendum.html

39 Siehe im BGHZ95, 274 (283), aber auch bei Hertin (1998, Rn. 6) und Schack (2005, Rn. 535).

40 Ausführlich dazu Mantz (2006*c*, S. 78 f.).

Verlag einräumbare Nutzungsrecht. Die Unterschiede, und in der Folge auch die möglichen Rechtsunsicherheiten, von Open Access gegenüber Open Content sind mithin vielfältig.

5 Zusammenfassung der rechtlichen Gemeinsamkeiten und Unterschiede

Als wichtigste Gemeinsamkeit der verschiedenen hier betrachteten Lizenzen ist das Copyleft-Prinzip zu nennen. Open Access kann zwar grundsätzlich auch ohne Copyleft realisiert werden, je nach Verständnis des Begriffs Open Access könnte auch die kostenfreie Verfügbarkeit auf dem Server eines Verlages darunter fallen. Allerdings zeigt doch die Konstruktion der *Digital Peer Publishing License*, dass zumindest bei ausdrücklicher Lizenzgestaltung das Konzept des Copyleft von Open Source und Open Content übernommen wurde.

Auch der Vertragsschluss ist bei allen Alternativen gleich gestaltet, schließlich ähneln sich die Lizenzen sehr. Die Rechte werden bei den Lizenzen fast immer durch den Rückfall an den Urheber bei einer Vertragsverletzung abgesichert, was im deutschen Recht über eine auflösende Bedingung realisiert wird. Unterschiede ergeben sich insbesondere bei der Modularität der Lizenzen. So erlauben Open-Content- und Open-Access-Lizenzen das Verbot von Bearbeitungen und kommerzieller Nutzung. Aus diesen Wahlmöglichkeiten hinsichtlich der Vertragsmodalitäten resultieren jedoch Abgrenzungsprobleme. Je stärker hiervon Gebrauch gemacht wird, desto mehr verlagern sich die rechtlichen Probleme. Während bei Open Source und Open Content häufig flexible Urhebergemeinschaften auftreten, basiert Open Access in vielen Fällen nicht auf einer Bildung von Gruppen. Vielmehr sollen Diskurs und Weiterentwicklung in der Wissenschaft gefördert werden. Dadurch stellen sich Probleme mit Gesellschaften und der Feststellung von Urheberrechtsinhabern nicht.

Während bei Open Content Verwertungsgesellschaften eine Rolle spielen können, sind bei Open Access zusätzlich die Verlage eine wichtige Größe, die noch immer maßgeblich auf die Vertragsgestaltung und die Möglichkeiten der Urheber einwirkt. Dieser Einfluss kann durch die Auflösung der Monopolisierung bzw. Oligopolisierung der anerkannten Vertriebswege verringert werden. Gerade diesen Weg schlagen Open-Access-Zeitschriften ein.

6 Fazit

Open Source, Open Content und Open Access stehen sich in vielerlei Hinsicht sehr nahe. Alle drei Konzepte dienen der Verfolgung eines gemeinsamen, übergeordneten Ziels: der freien Verfügbarkeit und Nutzbarkeit des Gutes *Information*. Als Folge ähneln sich die Motivationswege. Aber auch die Werkzeuge der Rechtssicherung haben viele Eigenschaften gemein.

Aus der Vielgestaltigkeit der Werke und den Entwicklungshintergründen ergeben sich jedoch einige Unterschiede, die sich auch in den verwendeten Lizenzverträgen niedergeschlagen haben. In der rechtlichen Behandlung muss dementsprechend, trotz der Gemeinsamkeiten, differenziert werden. Die rechtlichen Konstrukte sind nicht ohne Weiteres übertragbar. Vielmehr sind jeweils eine detaillierte Analyse der Rechtsverhältnisse und Fingerspitzengefühl erforderlich, um den tatsächlichen und historischen Unterschieden auch in rechtlicher Hinsicht gerecht zu werden.

Spannend bleibt, wie sich die Situation in Zukunft entwickeln wird. Hier sind bei Open Source und Open Content zum einen natürlich die Lizenzänderungsbemühungen bei der *GPL* bzw. den Creative-Commons-Lizenzen zu beachten, wobei auch die Akzeptanz der neuen Lizenzmodelle in der jeweiligen Nutzergemeinde eine wesentliche Rolle spielen wird. Auf der anderen Seite bleiben die Reaktionen der Rechteverwerter bzw. Verlage auf die neue Herausforderung der offenen Ressourcen im weiteren Sinne abzuwarten. Nicht zuletzt stehen insbesondere im Bereich des Urheberrechts in Deutschland einschneidende gesetzliche Änderungen an, die auch die Rechteverwertung bei Open Content und Open Access sowie den Zugang zu Informationen allgemein maßgeblich zu beeinflussen vermögen.

Literatur

Danowski, P. und Voß, J. (2005), Das Wissen der Welt – Die Wikipedia, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005', Lehmanns Media, Berlin, S. 393–405. http://www.opensourcejahrbuch.de [12. Dez 2006].

Diedrich, O. (2006), 'Deutsches Gericht bestätigt Gültigkeit der GPL', heise online. http://www.heise.de/newsticker/meldung/78530 [12. Dez 2006].

Dong, P., Loh, M. und Mondry, A. (2005), 'Relevance Similarity: An Alternative means to monitor Information Retrieval Systems', *Biomedical Digital Libraries* **2**(6). doi:10.1186/1742-5581-2-6, http://www.bio-diglib.com/content/2/1/6 [12. Dez 2006].

Grassmuck, V. (2004), *Freie Software – Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn. http://freie-software.bpb.de/ [12. Dez 2006].

Grützmacher, M. (2006), 'In der Regel keine Gesellschaft', *ITRB – Der IT-Rechts-Berater* S. 108.

Herrmann, T. und Kösch, S. (2004), 'Das GEMA Kuddelmuddel', *DE:BUG* **83**. http://www.de-bug.de/texte/3392.html [12. Dez 2006].

Hertin, P. W. (1998), Vorbemerkung vor § 31, *in* W. Nordemann, F. K. Fromm, K. Vinck und P. W. Hertin (Hrsg.), 'Urheberrecht – Kommentar zum Urheberrecht und zum Urheberrechtswahrnehmungsgesetz', 9. Aufl., Kohlhammer, Stuttgart.

Heussen, B. (2000), Sklaven, Bücher und Ideen – Rechtliche Chancen und Gefahren des E-Commerce, *in* C. Schertz und H. J. Omsels (Hrsg.), 'Festschrift für Paul W. Hertin zum 60. Geburtstag', C. H. Beck, München.

Hilty, R. M. (2006), 'Das Urheberrecht und der Wissenschaftler', *GRUR Int.* **55**(3), S. 179–190.

Jaeger, T. (2006), 'Eine Lizenz entsteht – Neue Entwürfe für GPL und LGPL', heise open. http://www.heise.de/open/artikel/76248 [12. Dez 2006].

Jaeger, T. und Metzger, A. (2006), *Open Source Software. Rechtliche Rahmenbedingungen der Freien Software*, 2. Aufl., C. H. Beck.

Jaeger, T. und Schulz, C. (2005), Gutachten zu ausgewählten rechtlichen Aspekten der Open Source Software, Gutachten, Institut für Rechtsfragen der Freien und Open Source Software (ifrOSS). http://www.ifross.de/ifross_html/art47.pdf [12. Dez 2006].

John, U. (1984), 'Grundsätzliches zum Wirksamwerden empfangsbedürftiger Willenserklärungen', *AcP* **184**, S. 385.

Knauf, S. (2006), Schutz von Open Access-Dokumenten und Datenbanken, *in* G. Spindler (Hrsg.), 'Rechtliche Rahmenbedingungen von Open Access-Publikationen', Universitätsverlag Göttingen, Göttingen, S. 105–154.

Knoblich, W. (2006), Erfolgreich mit Open Source – Das Red-Hat-Open-Source-Geschäftsmodell, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 155–164. http://www.opensourcejahrbuch.de.

Koch, F. (2000), 'Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software', *Computer und Recht* **16**(5), S. 273–281.

Liang, L. (2004), A Guide to Open Content Licences, Studie, Piet Zwart Institute, Willem de Kooning Academy, Hogeschool Rotterdam. http://pzwart.wdka.hro.nl/mdr/pubsfolder/opencontentpdf [12. Dez 2006].

Mantz, R. (2006*a*), 'Creative Commons-Lizenzen gerichtlich wirksam', *Computer und Recht* (5), S. R55.

Mantz, R. (2006*b*), 'GPL: Version 3 zur Diskussion', *Computer und Recht* (4), S. R42 f.

Mantz, R. (2006*c*), Open Access-Lizenzen und Rechtsübertragung bei Open Access-Werken, *in* G. Spindler (Hrsg.), 'Rechtliche Rahmenbedingungen von Open Access-Publikationen', Universitätsverlag Göttingen, Göttingen, S. 55–103.

Martens, S. (2006), 'LG Frankfurt/M.: Gültigkeit der GPL', *ITRB – Der IT-Rechts-Berater* **11**, S. 245.

Merten, S. und Meretz, S. (2005), Freie Software und Freie Gesellschaft, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005', Lehmanns Media, Berlin, S. 293–309. http://www.opensourcejahrbuch.de [12. Dez 2006].

Moglen, E. (2006), GPL v3 – Die Diskussion ist eröffnet, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 263–266.

Möller, E. (2006), Freiheit mit Fallstricken: Creative-Commons-NC-Lizenzen und ihre Folgen, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 271–282. http://www.opensourcejahrbuch.de [12. Dez 2006].

Neumann, J. (2006), Auf dem Weg zu einem Open-Access-Geschäftsmodell – Erfahrungsbericht German Medical Science, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 319–336.

Over, A., Maiworm, F. und Schelewsky, A. (2005), Publikationsstrategien im Wandel – Ergebnisse einer Umfrage zum Publikations- und Rezeptionsverhalten unter besonderer Berücksichtigung von Open Access, Studie, Deutsche Forschungsgemeinschaft, Bonn. http://www.dfg.de/dfg_im_profil/zahlen_und_fakten/statistisches_berichtswesen/open_access/download/oa_ber_dt.pdf [12. Dez 2006].

Pflüger, T. und Ertmann, D. (2004), 'E-Publishing und Open Access: Konsequenzen für das Urheberrecht im Hochschulbereich', *Zeitschrift für Urheber- und Medienrecht* **48**, S. 436–443.

Rasch, C. (2000), 'A Brief History of Free/Open Source Software Movement'. http://www.cisnet.com/glennmcc/foss/brief-open-source-history.html [12. Dez 2006].

Raymond, E. S. (1999), The Cathedral and the Bazaar, *in* 'The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary', O'Reilly & Associates, Cambridge, MA, USA, Kapitel The Social Context of Open-Source Software.

Schack, H. (2005), *Urheber- und Urhebervertragsrecht*, Mohr Siebeck, Tübingen.

Schricker, G. (2006), Vor §§ 28 ff. Vorbemerkung, *in* G. Schricker (Hrsg.), 'Urheberrecht', 3. Aufl., C. H. Beck, München, S. 565–622, Rn. 63.

Sester, P. (2000), 'Open-Source-Software: Vertragsrecht, Haftungsrisiken und IPR-Fragen', *Computer und Recht* **16**(12), S. 797–807.

Sietmann, R. (2006), 'Über die Ketten der Wissensgesellschaft', *c't* (12), S. 190.

Smid, V. (2006), Novell goes Open Enterprise, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 19–30.

Spindler, G. (2004), *Rechtsfragen bei Open Source*, Otto Schmidt, Köln.

Spindler, G. (2005), Miturhebergemeinschaft und BGB-Gesellschaft, *in* A. Ohly, T. Bodewig, T. Dreier, H.-P. Götting, M. Haedicke und M. Lehmann (Hrsg.), 'Perspektiven des Geistigen Eigentums und Wettbewerbsrechts – Festschrift für Gerhard Schricker zum 70. Geburtstag', C. H. Beck, München, S. 539–558.

Stürmer, M. und Myrach, T. (2006), Open Source Community Building, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006', Lehmanns Media, Berlin, S. 219–234. http://www.opensourcejahrbuch.de [12. Dez 2006].

Sujecki, B. (2005), 'Vertrags- und urheberrechtliche Aspekte von Open Source Software im deutschen Recht'. http://www.jurpc.de/aufsatz/20050145.htm [25. Nov 2005].

Waidele, S. (2006), 'Kreatives Schaffen durch Teilen'. http://stefan.waidele.info/2006/05/09/kreatives-schaffen-durch-teilen [12. Dez 2006].

Wandtke, A.-A. und Grunert, E. W. (2006), Vorbemerkung vor §§ 31 ff., *in* A.-A. Wandtke und W. Bullinger (Hrsg.), 'Praxiskommentar zum Urheberrecht', C. H. Beck, S. 373–410, Rn. 47.

Open Access vs. DRM?

THOMAS DREIER UND KENDRA STOCKMAR

(CC-Lizenz siehe Seite 563)

Auf den ersten Blick widersprechen sich Open Access (OA) und Digital Rights Management (DRM): Die Open-Access-Bewegung will für die Nutzer den Zugang zu wissenschaftlichen Informationen möglichst offenhalten. Deren Anbieter hingegen versuchen mit DRM die Nutzungsintensität zu kontrollieren, um über entsprechende Preismodelle die Kosten für Aufbereitung und Angebot zu amortisieren. Entsprechend hart fällt die Kritik des einen am jeweils anderen Modell aus: Aus der Sicht von DRM erodiert OA die Grundlage für die Produktion wissenschaftlicher Informationen. Umgekehrt stellt DRM aus der Sicht der Open-Access-Bewegung eine bedrohliche Einschränkung von Wissenschaft, Forschung und Innovation dar. Der vorliegende Artikel stellt Philosophien und Wirkungsweisen von OA und DRM näher vor und versucht, die Möglichkeit einer Synthese aufzuzeigen.

Schlüsselwörter: Produktdiversifizierung · Micropayment · Archivierung · Urheberrechtsreform · Open-Access-Geschäftsmodell

1 Einleitung

Seit einiger Zeit haben Urheber und Rechteinhaber – unter ihnen auch die Verleger wissenschaftlicher, technischer und medizinischer Literatur (STM) – begonnen, auf die ihnen entgleitende Kontrolle der Verbreitung und Nutzung ihrer Verlagsprodukte mit technischen Schutzmaßnahmen zu reagieren. Dabei handelt es sich vor allem um Zugangs- und Kopiersperren. Ging es dabei zunächst vornehmlich um die Bekämpfung von Piraterie, so ermöglichen technische Schutzmechanismen jedoch auch unterschiedliche Vertriebsmöglichkeiten und das Management von Rechtebündeln unterschiedlicher Nutzergruppen (Guggemos 2004, S. 184).

Vor allem bei wissenschaftlichen Zeitschriften könnte das jedoch die Zugriffsmöglichkeiten der Wissenschaftler in einer Weise einschränken, die der Forschung und

Innovation abträglich ist. Schon ohne DRM ist es auf dem Markt der wissenschaftlichen Zeitschriften zu einer *journal crisis* gekommen. Angesichts der wissenschaftlichen Spezialisierung wächst die Zahl wissenschaftlicher Zeitschriften rascher als die Anschaffungsetats von Fachbibliotheken, wenn Letztere angesichts knapper öffentlicher Kassen überhaupt noch Zuwachsraten verzeichnen. So können längst nicht mehr alle Bibliotheken alle Zeitschriften halten. Es sinkt folglich die Zahl der Abonnenten, was wiederum eine Erhöhung der Preise zur Folge hat, die ihrerseits von den Bibliotheken nicht mehr mitgetragen werden kann. Kurzum: ein Teufelskreis.

DRM, so wird von Seiten der Wissenschaftler befürchtet, könnte diese Tendenz des zunehmend erschwerten Zugangs zu veröffentlichten wissenschaftlichen Informationen noch verstärken. DRM erschwert nicht nur den Zugang zu veröffentlichtem Material. DRM kann sogar an sich rechtmäßige Nutzungen verhindern, die der Gesetzgeber im Rahmen urheberrechtlicher Schrankenregelungen – wie etwa der Privatkopie – für zulässig erklärt hat. So herrscht in Teilen der Wissenschaftlerkreise der Eindruck vor, dass DRM Inhalte eher wegschließt als zugänglich macht, und dass Verlage, denen im wissenschaftlichen Bereich häufig eine Monopolstellung zukommt,[1] den Zugang nur unter finanziell inakzeptablen Bedingungen gewähren, die letztlich nur der Gewinnmaximierung dienen.[2]

In diesem Beitrag wird dagegen die These aufgestellt, dass OA und DRM trotz ihrer gegensätzlichen rechtspolitischen Grundausrichtung dennoch nicht unvereinbar nebeneinander stehen. Wahrscheinlicher dürfte in Zukunft vielmehr ein Szenario der zumindest teilweisen gegenseitigen Ergänzung sein. Dafür sind schon jetzt erste Anhaltspunkte erkennbar.

2 Open Access und DRM: Zwei gegensätzliche Philosophien

Um die eben genannte These der Vereinbarkeit von OA und DRM näher darzulegen, seien in diesem Teil zunächst die Philosophien von DRM (Abschnitt 2.1) und von OA (Abschnitt 2.2) kurz skizziert, ehe Teil 3 die Möglichkeiten einer Ergänzung anhand sowohl eines Praxisbeispiels wie auch aktueller Gesetzgebungsvorhaben auslotet (Abschnitt 3.1 bzw. 3.2).

2.1 DRM – Barriere oder Chance?

Seit dem Aufkommen digitaler Vervielfältigungsmöglichkeiten fürchten Urheber und die Inhaber von Rechten an urheberrechtlich geschützten Werken und Leistungen, dass ihre analogen wie digitalen Werke unerlaubt kopiert und weiterverbreitet werden. In Zeiten der Vernetzung sehen die Rechteinhaber ihre Kontrollmöglichkeiten noch um ein Vielfaches weiter schwinden.

1 Vgl. zur Konzentration auf dem Zeitschriftenmarkt etwa die Studie von Dewatripont et al. (2006).
2 So auch Hilty (2006, S. 184).

So sind viele Rechteinhaber schon seit geraumer Zeit dazu übergegangen, den Zugang und/oder die Art und Weise der Nutzung ihrer Werke und Leistungen durch technische Mittel zu kontrollieren (sog. technische Schutzmechanismen, TSM, englisch: *technical protection measures*, TPM). Schlagworte wie „self-help in the digital jungle" (Dam 2001) oder die zum geflügelten Wort gewordene Wendung „the answer to the machine is in the machine" (Clark 2005) verdeutlichen diese Strategie. Nun sind solche technischen Zugangssperren und Nutzungseinschränkungen aber ebenfalls nicht gegen ihre Umgehung gefeit. Es besteht ein wohl nie endender Wettlauf zwischen der Entwicklung und Anwendung technologischer Schutzmaßnahmen und deren Umgehung. Wenn und soweit der Einsatz technischer Schutzmechanismen seitens der Rechteinhaber jedoch legitim ist, um die gesetzlich gewährte Ausschließlichkeit angesichts digitaler und vernetzter Technologie aufrechtzuerhalten, ist der Gesetzgeber in einer gewissen Pflicht, gegen die unerlaubte Umgehung technischer Schutzmechanismen wiederum einen rechtlichen Umgehungsschutz bereitzustellen.[3]

Es geht bei technischen Schutzmechanismen allerdings um mehr als nur um den technischen Schutz urheberrechtlich geschützter Werke und Leistungen gegen die unbefugte Nutzung. Zum einen lassen sich mit technischen Mitteln bislang analoge Transaktionen digital und mithin weitgehend automatisiert vornehmen. Das ermöglicht das Digital Rights Management. So können etwa über das Netz fremde geschützte Werke lokalisiert, angefordert, abgerufen, übermittelt und schließlich genutzt werden, wobei der Vorgang protokolliert wird und die protokollierten Transaktionsdaten zur Abrechnung genutzt werden.

Darüber hinaus ermöglichen technische Schutzmechanismen über die bloße Pirateriebekämpfung hinaus auch eine Produktgestaltung mit abgestuften Nutzungsberechtigungen (sog. Produktdiversifizierung). Ein Beispiel für eine solche Diversifizierung wäre es, einen Aufsatz oder ein Buch einmal ohne jeglichen Kopierschutz, mit einer bestimmten Anzahl von erlaubten Kopier- bzw. Druckvorgängen, mit oder ohne Bindung an ein bestimmtes Endgerät oder aber lediglich zur Ansicht ohne Druckfunktion zu vermarkten. Eine derartige Produktdiversifizierung ermöglicht im Weiteren eine Preisdifferenzierung, die ihrerseits wiederum zu einer besseren Abschöpfung der Marktnachfrage und letztlich zu einer besseren Versorgung der Konsumenten führt.[4] Technische Schutzmechanismen verhindern also nicht nur, dass urheberrechtlich geschützte Werke und Leistungen von dazu überhaupt nicht Befugten genutzt werden, sondern sie können zugleich sicherstellen, dass die technisch geschützten Werke und Leistungen von denjenigen, die zur Nutzung in einem bestimmten Umfang berechtigt sind, nicht in einem weitergehenden Umfang genutzt werden.

3 Vgl. zum Ganzen vor allem Bechtold (2002) sowie zuvor Wand (2001).

4 Das ist allerdings nicht unumstritten. Vgl. nur Einhorn (2001, 2002) zum einen und Benkler (1999, S. 377–384) zum anderen.

2.2 Open Access: Philosophie und Modelle

Der Begriff *Open Access* bezieht sich demgegenüber auf den freien Zugang zu wissenschaftlicher Literatur in elektronischer Form, auf die über das Internet von überallher zugegriffen werden kann. Bei der Verwirklichung von Open Access kann generell zwischen zwei Hauptmodellen unterschieden werden: Beim ersten Modell handelt es sich um (Primär-)Veröffentlichungen in Open-Access-Zeitschriften (*open access journals*, OAJ), die für jedermann gebührenfrei zugänglich sind und sich aus verschiedenen Quellen finanzieren (sog. „Golden Road"). Die andere Open-Access-Form ist das *archiving* (*open access archiving*, OAA) in unterschiedlichen Ausprägungen (sog. „Green Road"). Dabei werden Artikel, die bereits veröffentlicht bzw. bei einer Zeitschrift akzeptiert wurden, zusätzlich in elektronischer Form in einem Archiv bzw. auf der persönlichen Website des Autors öffentlich zugänglich gemacht.

Aber auch über diese Unterscheidung hinaus erfährt der Begriff des *freien* Zugangs unterschiedliche Auslegungen. Besonders umstritten ist, ob Open Access tatsächlich einen in jeder Hinsicht kostenfreien Zugang bedeuten muss. So nimmt etwa die *Budapest Open Access Initiative* einen freien Zugang nur dann an, wenn neben den anfallenden Kosten für den Internetzugang selbst keine weiteren Gebühren für den Zugriff auf die Literatur zu entrichten sind (Open Society Institute 2002). Dagegen heißt es in der *Göttinger Erklärung zum Urheberrecht für Bildung und Wissenschaft* etwas weniger kategorisch: „Freier Zugang zu Information und Wissen muss nicht vergütungsfrei bedeuten" (Kuhlen et al. 2004).

Neben der Frage der Zugangsgebühren geht es auch um die weitere Frage, was mit den Open-Access-Werken geschehen darf, also welche Nutzungsrechte daran bestehen sollen. Neben den *price barriers* geht es also auch um *permission barriers* (Suber 2004). Auch hier besteht jedoch noch keine in jeder Hinsicht einheitliche Terminologie und auch in Bezug auf die Nutzungsrechte, die zur Verwirklichung von Open Access erforderlich sind, besteht gegenwärtig noch keine einheitliche Auffassung.

Darüber hinaus fordert etwa die *Berliner Erklärung*, dass Open-Access-Publikationen in Online-Archiven bereitgehalten werden müssen, die gewisse technische Standards erfüllen (Max-Planck-Gesellschaft 2003). Vor allem wird Wert auf Interoperabilität wie auch auf die Sicherstellung der Langzeitarchivierung gelegt. Dazu müssen auch die Publikationen selbst mitsamt ihrer Metadaten in einem angemessenen Format gespeichert sein.

Die *Wiener Erklärung zum freien Informationszugang* enthält demgegenüber zwar keine Definition des Begriffs Open Access, beschreibt aber zehn Thesen zur Informationsfreiheit (Forgó et al. 2005). Darunter findet sich vor allem die Forderung nach einem vernünftigen Gleichgewicht zwischen den Interessen von Autoren, Institutionen, Verwertern und der Allgemeinheit. Ein Mittel hierzu sehen die Autoren hauptsächlich in einer Anpassung der rechtlichen und sozialen Rahmenbedingungen, wobei „das Recht [. . .] Vorrang zu genießen [hat] vor technisch bedingten Tatsachen,

die von beherrschenden Marktteilnehmern geschaffen werden."

Damit bringt die *Wiener Erklärung* die DRM-Problematik ausdrücklich in Zusammenhang mit den Zielen von OA und bezieht in diesem Punkt eindeutig Stellung. Es wird die Durchsetzbarkeit von Rechten auch gegenüber technischen Schutzmaßnahmen gefordert und gleichzeitig die Gefahr des Missbrauchs durch mächtige Marktteilnehmer, wie etwa großen Wissenschaftsverlagen, angesprochen. Der Einsatz von DRM scheint demnach also dem OA-Gedanken fundamental zu widersprechen.

3 DRM und Open Access: Eine Synthese?

Dennoch erscheint eine Synthese dieser beiden unterschiedlichen Ansätze durchaus denkbar. Denn DRM kann auch für die Verwirklichung von Open-Access-Modellen eingesetzt werden. Zum einen setzt OA nicht zwangsweise voraus, dass Inhalte gänzlich kostenfrei zugänglich sind, und zum anderen muss DRM nicht notwendigerweise nur bei entgeltpflichtigen Angeboten zum Einsatz kommen. Schon jetzt kann in der Praxis eine wenn auch langsame, so doch immerhin teilweise Annäherung der beiden Konzepte beobachtet werden.

So macht DRM insbesondere im Bereich der Archivierung (OAA) durchaus Sinn. DRM vermag hier vor allem Interoperabilität und Durchsuchbarkeit zu gewährleisten. Sofern bei OA unterschiedliche Zugangsmodelle („Golden Road", „Green Road") implementiert werden, bietet sich auch hier die Möglichkeit für einen unterstützenden Einsatz durch DRM.

Auch im Rahmen der Überarbeitung der Creative-Commons-Lizenzen zur Veröffentlichung von Open Content wird eine mögliche Änderung diskutiert. Danach soll die bisherige Regelung, dass gegen die Lizenz verstoßen wird, wenn ein Werk mit technischen Schutzmaßnahmen versehen ist, dahingehend modifiziert werden, dass die Lizenzbedingungen auch dann erfüllt sein sollen, solange gleichzeitig eine nicht geschützte Version frei zugänglich ist. Dieser Vorschlag wurde jedoch letztendlich nicht angenommen.[5]

Ein Beispiel aus der Praxis für den Einsatz von DRM in einem Geschäftsmodell für OA-Journale bietet letztlich die wissenschaftliche Zeitschrift *Cell Science* (Abschnitt 3.1). Schließlich sind in Bezug auf die Archivierung bereits gesetzliche Lösungsmodelle formuliert worden, die eine sinnvolle Nutzung von DRM-Technologien im Rahmen des OAA einbeziehen (Abschnitt 3.2).

3.1 Die Zeitschrift Cell Science: Ein Open-Access-Geschäftsmodell unter Verwendung von DRM in der Praxis

Auch elektronische OA-Zeitschriften, die ihre Inhalte frei zur Verfügung stellen, sind gezwungen ihre Kosten zu decken. Dafür stehen unterschiedliche Geschäftsmodelle

5 Eine Übersicht zum Verlauf der Diskussion geben Gehring (2006) sowie Knauf (2006).

zur Auswahl. Denkbar sind etwa staatliche Subventionen. Bei einem Autorengebührenmodell werden anstelle der sonst üblichen Subskriptionsgebühren, die von den Beziehern der Zeitschriften entrichtet werden, Publikationsgebühren von den Autoren erhoben. Auch Mischmodelle sind denkbar.

Vorliegend ist vielmehr von Interesse, dass ein auf dem Prinzip des Pay-per-View basierendes Vergütungsmodell, bei dem der Nutzer jeweils für den Zugang zur Ansicht einzelner Artikel oder einer einzelnen Zeitschrift bezahlt, im Ergebnis durchaus zu einer größeren Zugänglichkeit der einzelnen Artikel bzw. der Zeitschriften insgesamt führen kann. In der Tat werden derartige Modelle bereits erprobt. Voraussetzung dafür ist zum einen das Vorhandensein entsprechender Technologien, insbesondere von Digital-Rights-Management-Systemen. Zum anderen muss es möglich sein, dass die Abrechnung über Micropayment erfolgen kann. Micropayment beschreibt ein Verfahren, bei dem auch kleinere und Kleinstbeträge online bezahlt werden können.[6] Entscheidend für die Nutzung solcher Systeme sind ausreichend geringe Transaktionskosten, die bei herkömmlichen Online-Bezahlverfahren wie bspw. dem gängigen Kreditkartensystem zu hoch wären,[7] beim Mobiltelefon jedoch bereits möglich sind.

Nach diesem Modell hat die STM-Zeitschrift *Cell Science* im Mai 2006 erstmals ein solches Micropayment in ihr Geschäftsmodell integriert. Interessant ist dabei, dass die „Minigebühren" nicht allein an die Zeitschrift fließen sollen, sondern sogar als Tantiemen für die Autoren gedacht sind.[8] Zielsetzung ist dabei, die Kosten für Leser und Institutionen zu reduzieren und gleichzeitig die Autoren in höherem Maße als bislang weithin üblich zu beteiligen. Nach Ansicht von *Cell Science* könnte Micropayment eine Möglichkeit darstellen, den Umsatz einer Zeitschrift an die tatsächliche Nachfrage zu koppeln, was bei den momentan gängigen subskriptionsbasierten Zeitschriften wenn überhaupt, so allenfalls über den Umweg der Reputation erfolgt. Ein derartiges DRM-unterstütztes Zugangssystem hätte auch einen Vorteil gegenüber den heute üblichen Konsortiallizenzen, bei denen große Bibliotheken, wenn nicht gar Bibliotheksverbünde, ganze Bündel von Zeitschriften abonnieren und meist gegen einen nur vergleichsweise geringen Aufpreis auch den elektronischen Zugriff auf die abonnierten Zeitschriften erhalten. Denn zugriffsberechtigt sind bei solchen Konsortiallizenzen immer nur die Nutzer der betreffenden Bibliotheken, im Universitätsbereich also nur die Universitätsangehörigen. Hingegen sind etwa Absolventen, die sich selbständig machen, von der Nutzung abgeschnitten und ihnen ist angesichts der für sie jeweils zu großen Bündel mithin auch der Zugriff auf einzelne Artikel aus finanziellen Gründen praktisch verwehrt.

Freilich mag man einwenden, dass ein solches auf DRM und Micropayment basierendes Pay-per-View-Modell nicht mehr den Anforderungen entsprechen könnte,

6 Einen Überblick über verschiedene Online-Bezahlsysteme und aktuelle Entwicklungen bietet bspw. Lammer (2005, S. 343–362).

7 Siehe etwa Párhony et al. (2005) mit weiteren Nachweisen.

8 Siehe dazu Suber (2006) und auch Cell Science (2006).

welche die einzelnen OA-Erklärungen und -Definitionen stellen. Ob man hier noch von *Open Access* sprechen kann, dürfte letztlich jedoch vor allem von der Höhe der Gebühr und der konkreten Ausgestaltung der Zugangsbedingungen sowie den sonstigen Vorteilen, wie insbesondere der Verfügbarkeit von Metadaten, abhängen.

Nicht der Einsatz oder Nichteinsatz von DRM entscheidet also über die Frage des Grades der Zugänglichkeit, sondern die Festlegung des Preises für das Einzeldokument. Zugleich erweist sich ein Festhalten an der Forderung eines gänzlich kostenfreien Zugangs im Hinblick auf das Ziel eines möglichst umfassenden Zugangs letztlich als kontraproduktiv, schließt es doch Modelle aus, die wie das auf DRM aufsetzende Pay-per-View-Verfahren unter bestimmten Randbedingungen einen im Vergleich zum bisherigen System erhöhten Zugang versprechen.

3.2 DRM beim open access archiving? Einsatzmöglichkeiten vor dem Hintergrund aktueller Gesetzesvorschläge

In letzter Zeit wird das Thema der Förderung von OA zu wissenschaftlicher Literatur in Deutschland im Rahmen der gegenwärtigen Urheberrechtsreform diskutiert und zu diesem Zweck eine Änderung des geltenden Urheberrechtsgesetzes vorgeschlagen (siehe nächster Unterabschnitt). Inwiefern Digital Rights Management eine sinnvolle Ergänzung bei einer etwaigen Umsetzung der Vorschläge sein kann, soll abschließend kurz ausgelotet werden (siehe Unterabschnitt „DRM im Rahmen gesetzlicher Regelungen zur Förderung von OA“).

Die unterschiedlichen Vorschläge zur Gesetzesänderung

Nach deutschem Urheberrecht ist Urheber eines wissenschaftlichen Werkes immer der Wissenschaftler, der es verfasst hat. Handelt es sich dabei um einen wissenschaftlichen Hochschulangestellten, der das fragliche Werk in Erfüllung seiner Verpflichtungen aus dem Angestelltenverhältnis geschaffen hat, so stehen die Nutzungsrechte grundsätzlich der Hochschule zu. Anders hingegen verhält es sich bei Professoren und Assistenten, deren eigene Forschungstätigkeit von Art. 5 Abs. 3 des Grundgesetzes (GG) geschützt ist und die zur Veröffentlichung grundsätzlich nicht verpflichtet sind.[9] Das führt in der Praxis zumeist dazu, dass die Professoren und Assistenten ihre Rechte den Verlagen exklusiv für deren Publikationen übertragen.

Um hier im Sinne von Open Access Abhilfe zu schaffen, wurde im Jahr 2004 von Seiten der Baden-Württembergischen Ministerialbürokratie zunächst vorgeschlagen, den Hochschulen an den von allen ihrer Beschäftigten erstellten Schriftwerken ein Erstveröffentlichungsrecht einzuräumen bzw. ihnen insoweit zumindest eine Anbietungspflicht aufzuerlegen (Pflüger und Ertmann 2004, S. 436). Der genaue Wortlaut

9 Vgl. Bundesgerichtshof (1991, S. 525) sowie zum Ganzen Kraßer und Schricker (1988) sowie die weiteren Nachweise bei Dreier und Schulze (2006, § 43 Rdnr. 12).

einer gesetzlichen Regelung blieb danach zwar noch offen, doch sollte ein neu eingefügter Abs. 2 von § 43 UrhG Hochschulbeschäftigte verpflichten, „ein im Rahmen [ihrer] Forschungstätigkeit entstandenes Werk nach § 2 Abs. 1 Nr. 1 UrhG der Hochschule – ggf. auch nicht exklusiv – zur Veröffentlichung anzubieten" (Pflüger und Ertmann 2004, S. 441). Dies würde bedeuten, dass die Hochschule zunächst das Recht hat, die Werke ihrer Beschäftigten, gegebenenfalls auch über eigene Medien, zu veröffentlichen. Lediglich im Fall der nicht-exklusiven Inanspruchnahme hätte der Urheber dann noch die Möglichkeit, sein Werk gleichzeitig oder zeitversetzt in einem anderen Verlag zu veröffentlichen. In der Praxis kommt es zu einer solchen Parallel- oder Zweitveröffentlichung allerdings nur dann, wenn der Verlag eine solche trotz der Erstveröffentlichung durch die Universität wirtschaftlich noch für sinnvoll hält.

Ein abgemilderter Vorschlag der Kultusministerkonferenz aus dem Jahr 2005 sucht ebenfalls an § 43 UrhG anzusetzen (Köcher et al. 2005), will den Universitäten jedoch nur ein nicht-ausschließliches Nutzungsrecht an bereits zur Veröffentlichung vorgesehenen Werken verschaffen (sog. Zweitveröffentlichungsrecht). Das soll es ermöglichen, dass jede Publikation, die in den Anwendungsbereich der Norm fällt, in einem online zugänglichen Archiv zum freien Download bereitgehalten werden kann. Das Nutzungsrecht der Hochschule soll danach auf eine nicht-kommerzielle Nutzung beschränkt sein und überdies nur die Online-Zugänglichmachung erfassen, nicht hingegen die Veröffentlichung über sonstige Medien.

Ein anderer, von einem Wissenschaftler der Max-Planck-Gesellschaft erarbeiteter Vorschlag verschiebt den Zeitpunkt, ab dem die Hochschulen auf die Veröffentlichungen ihrer Mitarbeiter zugreifen können sollen, so dass die Verlage zunächst einen gewissen Vorlauf (sog. *lead time*) haben, während dessen sie die wissenschaftlichen Werke allein veröffentlichen und vermarkten können. Gesetzestechnisch wird dazu in Ergänzung des bisherigen § 38 Abs. 1 UrhG vorgeschlagen, dass das Recht zur öffentlichen Zugänglichmachung an wissenschaftlichen Publikationen nach sechs Monaten zwingend an den Urheber zurückfällt (Hansen 2005). Im Ergebnis wird damit die dem Erstverleger zunächst gewährte Exklusivität in zeitlicher Hinsicht beschränkt, allerdings auch wiederum nur im Hinblick auf die öffentliche Zugänglichmachung, da das Ziel eines möglichst unbehinderten Online-Zugangs ja auch nur insoweit eine Einschränkung der Ausschließlichkeit der Rechte erfordert. Der Autor hätte nach Ablauf der sechs Monate also die Möglichkeit, seinen Artikel entweder selbst ins Netz zu stellen und/oder ihn einem Open-Access-Archiv zu überlassen. Diesen Vorschlag hat der Bundesrat in Ergänzung des ursprünglichen Regierungsentwurfs in seiner Stellungnahme zum 2. Korb der gegenwärtigen Urheberrechtsreform aufgegriffen (Bundesrat 2006, Punkt 6, S. 6).

Dieser Vorschlag hat aus OAA-Sicht jedoch den Nachteil, dass die einzelnen Urheber ihre veröffentlichten Artikel einem Open-Access-Archiv zwar überstellen können, dies jedoch nicht tun müssen. Aus OAA-Sicht wäre mit Stevan Harnad (2006) also noch zu fordern, dass die Autoren zumindest die Metadaten ihrer Artikel bei einem

Open-Access-Archiv einreichen, wenn nicht gar, dass die Autoren ihr Werk auch tatsächlich hinterlegen müssen, wobei die Entscheidung über eine Zugänglichmachung immer noch dem Autor überlassen bliebe.

DRM im Rahmen gesetzlicher Regelungen zur Förderung von OA

Es geht in diesem Beitrag nicht darum, die Stärken und Schwächen der genannten Ansätze, ihre politische Realisierbarkeit oder gar ihre verfassungsrechtliche Zulässigkeit zu überprüfen.[10] Im Mittelpunkt der Überlegungen steht vielmehr die Frage, inwieweit die soeben skizzierten Ansätze den Einsatz von DRM ermöglichen oder vielleicht sogar erfordern.

Bei der ersten Alternative (Erstveröffentlichungsrecht der Universitäten) könnten die einzelnen Hochschulen DRM einsetzen wollen, da sie damit ja in die Rolle der Verleger geschlüpft sind und sie für die Offenhaltung des Zugangs zu den von ihnen veröffentlichten oder vorgehaltenen Werken ein wirtschaftlich tragfähiges Geschäftsmodell entwickeln müssen. Ein Einsatz von DRM könnte hier etwa dem Pay-per–View-Modell des oben genannten Beispiels von *Cell Science* entsprechen.

Bei der zweiten Alternative (nicht-ausschließliches Nutzungsrecht an bereits zur Veröffentlichung vorgesehenen Werken) könnte DRM zum Einsatz kommen, um die nicht-kommerzielle Verwendung zu gewährleisten oder etwa eine Einschränkung auf bestimmte Nutzerkreise vorzunehmen. Da jegliche Differenzierung von Nutzungsrechten, die neben das Verlagsrecht treten, dazu beiträgt, die Interessenlage besser auszugleichen, könnte DRM für eine entsprechende Differenzierung und deren Aufrechterhaltung sorgen. Das Setzen von Wasserzeichen könnte dabei die Wahrung des urheberpersönlichkeitsrechtlichen Namensrechts der Urheber schützen.[11] Nicht zuletzt vermögen DRM-Systeme die Ermittlung der Zitierhäufigkeit und das Management von Metadaten zu unterstützen (vgl. Suber 2003).

Bei der dritten Alternative (zeitlich begrenzter Rückfall der Online-Rechte an den Urheber) könnte DRM ebenfalls von Beginn an den gesamten Veröffentlichungsprozess unterstützen. Schon bei der Einreichung der Metadaten oder des Volltextes könnte der Autor festlegen, ob er einer Veröffentlichung nach sechs Monaten, oder ggf. auch später, zustimmen möchte. Daraufhin könnte der Inhalt vom automatisierten System nach der Zeitvorgabe ohne weiteres Zutun freigegeben werden. Anderenfalls könnte auch nach sechs Monaten eine automatisierte Anfrage des Systems über die Zustimmung zur Veröffentlichung, und/oder den Volltext die dieser leicht und schnell beantworten könnte. Darüber hinaus sind verschiedene andere Auswahlfunktionen vorstellbar, so etwa die Zugänglichmachung für bestimmte Nutzerkreise, die Verlinkung zur Verlagsseite oder zu ähnlichen oder weiterführenden Materialien.

10 Siehe dazu etwa Hansen (2005); Steinhauer (2006); Pflüger und Ertmann (2004); Köcher et al. (2005); Bundesrat (2006); Bundesregierung (2006).

11 Zur Identifizierung von Werken über Metadaten und Wasserzeichen siehe etwa Schulz (2006, S. 472).

Je komfortabler ein solches System und die entsprechenden Prozesse für den Autor ausgestaltet wären, desto mehr Zuspruch würde sich sicher für eine Zugänglichmachung in einem Open-Access-Archiv finden.

4 Fazit

Die vorstehenden Ausführungen sollten deutlich gemacht haben, dass OA und DRM in ihrem jeweiligen Ausgangspunkt zwar durchaus eine gegensätzliche Philosophie zu Grunde liegt. Dennoch stellen OA und DRM keine unversöhnlichen Gegensätze dar, sondern es kann DRM-Technologie durchaus zur Förderung von OA zum Einsatz kommen. Schon jetzt werden erste diesbezügliche Beispiele am Markt erprobt und erste Erfahrungen gesammelt. Das gilt umso mehr, wenn der deutsche Gesetzgeber den OA-Gedanken im Zuge der gegenwärtigen Urheberrechtsreform tatsächlich explizit durch eine Gesetzesänderung fördert.

Im Ergebnis ist das Verharren auf der unversöhnlichen Frontstellung zwischen DRM und OA wenig hilfreich. Vielmehr gilt es, Geschäftsmodelle zu entwickeln, bei denen DRM die OA-Ansätze und insbesondere die OA-Archivierung unterstützen kann. Welcher Gewinn an Zugangsmöglichkeiten damit verbunden sein wird, hängt im Ergebnis freilich nicht so sehr von der Frage ab, ob DRM eingesetzt wird oder nicht, sondern weit mehr davon, wie hoch der Preis für den Zugriff auf die im Einzelfall benötigten Dokumente sein wird. Hier aber spricht eine Vermutung dafür, dass dieser Preis dann, wenn der Zugriff durch technische Lösungen unterstützt wird, im Ergebnis eher sinken und nicht steigen wird.

Literatur

Bechtold, S. (2002), *Vom Urheber- zum Informationsrecht. Implikationen des Digital Rights Management*, Bd. 33 aus *Informationen und Recht*, C. H. Beck, München. http://www.jura.uni-tuebingen.de/bechtold/pub/2002/DRM_Diss.pdf [28. Jan 2007].

Benkler, Y. (1999), 'Free as the Air to Common Use: First Amendment Constraints on Enclosure of the Public Domain', *New York University Law Review* **74**(2), S. 354–446. http://www.yale.edu/lawweb/jbalkin/telecom/benklerfreeastheairtocommonuse.pdf [29. Jan 2007].

Bundesgerichtshof (1991), 'Grabungsmaterialien', *GRUR* **1991**(7), S. 523–529. Urteil vom 27. 09. 1990 I ZR 244/88.

Bundesrat (2006), 'Stellungnahme zum Entwurf eines Zweiten Gesetzes zur Regelung des Urheberrechts in der Informationsgesellschaft', BR-Drucksache 257/06 (Beschluss). http://www.bundesrat.de/cln_050/SharedDocs/Drucksachen/2006/0201-300/257-06_28B_29,templateId=raw,property=publicationFile.pdf/257-06(B).pdf [29. Jan 2007].

Bundesregierung (2006), 'Gegenäußerung zur Stellungnahme des Bundesrates zum Entwurf eines Gesetzes zur Regelung des Urheberrechts in der Informationsgesellschaft'. http://www.urheberrecht.org/topic/Korb-2/bmj/1250.pdf [29. Jan 2007].

Cell Science (2006), 'The Academic Library Funding Crisis – Micropayments as an Alternative to Subscriptions or Pay-to-publish'. http://www.cellscience.com/STM.html [17. Dez 2006].

Clark, C. (2005), The Future of Copyright in a Digital Environment, *in* T. D. und Jon Bing (Hrsg.), '"The Answer to the Machine is in the Machine" and other Collected Writings', Institutt for Rettsinformatikk, University of Oslo, S. 11 ff.

Dam, K. W. (2001), Self-Help in the Digital Jungle, *in* R. Dreyfuss, D. L. Zimmerman und H. First (Hrsg.), 'Expanding the Boundaries of Intellectual Property (Innovation Policy for the Knowledge Society)', Oxford University Press, New York, S. 103.

Dewatripont, M., Ginsburgh, V., Legros, P., Walckiers, A., Devroey, J.-P., Dujardin, M., Vandooren, F., Dubois, P., Foncel, J., Ivaldi, M. und Heusse, M.-D. (2006), Study on the economic and technical evolution of the scientific publication markets in Europe, Studie, DG-Research, Europäische Kommission.

Dreier, T. und Schulze, G. (2006), *Urheberrechtsgesetz. Kommentar*, C. H. Beck, München.

Einhorn, M. A. (2001), Digital rights management and access protection: An economic analysis, *in* J. C. Ginsburg (Hrsg.), 'ALAI Congress 'Adjuncts and Alternatives for Copyright", New York. http://www.alai-usa.org/2001_conference/pres_einhorn.doc [17. Dez 2006].

Einhorn, M. A. (2002), 'Digital Rights Management, Licensing, and Privacy'. http://ssrn.com/abstract=332720 [17. Dez 2006].

Forgó, N., Seböck, W., Reis, L., Philapitsch, F. und Nentwich, M. (2005), 'Wiener Erklärung zum freien Informationszugang: 10 Thesen zur Informationsfreiheit'. http://www.chaoscontrol.at/we.pdf [29. Jan 2007].

Gehring, R. A. (2006), 'Entwürfe für neue CC Lizenzen vorgelegt', Golem IT-News. http://www.golem.de/0611/48700.html [01. Nov 2006].

Guggemos, W.-C. (2004), 'Digital Rights Management im praktischen Einsatz', *Zeitschrift für Urheber- und Medienrecht* **48**(3), S. 183–188.

Hansen, G. (2005), 'Zugang zu wissenschaftlicher Information – alternative urheberrechtliche Ansätze', *GRUR Int* **54**(5), S. 378–388. http://www.gerd-hansen.net/Hansen_GRUR_Int_2005_378ff.pdf [29. Jan 2007].

Harnad, S. (2006), 'Generic Rationale and Model for University Open Access Self-Archiving Mandates'. http://eprints.ecs.soton.ac.uk/12078/ [13. Mar 2006].

Hilty, R. M. (2006), 'Das Urheberrecht und der Wissenschaftler', *GRUR Int* **55**(3), S. 179–190.

Knauf, S. (2006), Schutz von Open Access-Dokumenten und Datenbanken, *in* G. Spindler (Hrsg.), 'Rechtliche Rahmenbedingungen von Open Access-Publikationen',

Universitätsverlag Göttingen, Göttingen, S. 105–154. http://www.lehrstuhl-spindler.uni-goettingen.de/extern/openaccess/leitfaden/ [12. Dez 2006].

Kraßer, R. und Schricker, G. (1988), *Patentrecht und Urheberrecht an Hochschulen*, Nomos, Baden-Baden.

Kuhlen, R., Beger, G. und Degkwitz, A. (2004), 'Göttinger Erklärung zum Urheberrecht für Bildung und Wissenschaft', Aktionsbündnis „Urheberrecht für Bildung und Wissenschaft". http://www.urheberrechtsbuendnis.de/GE-Urheberrecht-BuW-Mitgl.pdf [29. Jan 2007].

Köcher, J. K., Bajon, B., Beger, G. und Höfinger, F. M. (2005), Stellungnahme zum Referentenentwurf für ein Zweites Gesetz zur Regelung des Urheberrechts in der Informationsgesellschaft vom 27. 09. 2004, *in* U. Sieber und T. Hoeren (Hrsg.), 'Urheberrecht für Bildung und Wissenschaft – Anforderungen an das Zweite Gesetz zur Regelung des Urheberrechts in der Informationsgesellschaft', Beiträge zur Hochschulpolitik 2/2005, Hochschulrektorenkonferenz, Bonn, S. 108–165. http://www.hrk.de/de/download/dateien/HRK%20-%20Reader%20Urheberrecht%202005.pdf [14. Jan 2006].

Lammer, T. (2005), *Handbuch E-Money, E-Payment & M-Payment*, Physica-Verlag, Heidelberg.

Max-Planck-Gesellschaft (2003), 'Berliner Erklärung über offenen Zugang zu wissenschaftlichem Wissen'. http://www.zim.mpg.de/openaccess-berlin/berlin_declaration.pdf [29. Jan 2007].

Open Society Institute (2002), 'Budapest Open Access Initiative – Initiativaufruf'. http://www.soros.org/openaccess/g/read.shtml [29. Jan 2007].

Pflüger, T. und Ertmann, D. (2004), 'E-Publishing und Open Access: Konsequenzen für das Urheberrecht im Hochschulbereich', *Zeitschrift für Urheber- und Medienrecht* **48**(6), S. 436–443.

Párhony, R., Nieuwenhuis, L. J. M. und Pras, A. (2005), The Fall and Rise of Micropayment Systems, *in* T. Lammer (Hrsg.), 'Handbuch E-Money, E-Payment & M-Payment', Physica Verlag, Heidelberg, S. 343 ff.

Schulz, D. (2006), 'Der Bedeutungswechsel des Urheberrechts durch Digital Rights Management – Paradigmenwechsel im deutschen Urheberrecht?', *GRUR* **55**(6), S. 470–477.

Steinhauer, E. W. (2006), 'Kritische Anmerkungen zum Zweitveröffentlichungsrecht nach § 38 Abs. 1 Satz 3 und 4 UrhG in der Fassung des Bundesratsentwurfes vom 9. Mai 2006', *Bibliotheksdienst* **40**(6), S. 734–742. http://www.zlb.de/aktivitaeten/bd_neu/heftinhalte2006/Recht010606.pdf [29. Jan 2007].

Suber, P. (2003), 'Removing barriers to research: An introduction to open access for librarians', *College & Research Libraries News* **64**(2). http://www.ala.org/ala/acrl/acrlpubs/crlnews/backissues2003/february1/removingbarriers.htm [29. Jan 2007].

Suber, P. (2004), 'Praising progress, preserving precision', *SPARC Open Access Newsletter* **77**. http://www.earlham.edu/~peters/fos/newsletter/09-02-04.htm#progress [29. Jan 2007].

Suber, P. (2006), 'Charging readers, paying authors, in micropayments', *Open Access News* **17. Mai 2006**. http://www.earlham.edu/~peters/fos/2006_05_14_fosblogarchive.html.

Wand, P. (2001), *Technische Schutzmaßnahmen und Urheberrecht*, C. H. Beck, München.

Ein Open-Content-Internetarchiv des Nachlasses eines zeitgenössischen Komponisten

KEI ISHII

(CC-Lizenz siehe Seite 563)

Der Artikel beschreibt den Aufbau eines Open-Content-Archivs aus dem Nachlass Maki Ishiis, eines bedeutenden japanischen Komponisten zeitgenössischer Musik. Die zeitgenössische Musik ist geprägt von einer intensiven direkten Kommunikation zwischen Komponisten und Musikern, wie fortlaufende Anfragen von Musikern nach Noten und anderen Materialien aus dem Nachlass belegen. Um den Zugang zu dem Nachlass auch für die Zukunft so einfach wie möglich zu gestalten, wurde entschieden, ein Open-Content–Archiv im Internet aufzubauen. Im Folgenden werden die Schritte zu einem solchen Archiv geschildert. Neben der Erstellung eines Werkverzeichnisses steht die Zugänglichmachung der nichtpublizierten Notenmanuskripte im Mittelpunkt der Betrachtung, und hier insbesondere die Festlegung der Lizenzbedingungen, um den Musikern eine möglichst hohe Rechtssicherheit zu gewährleisten. Die rechtlich-ökonomischen Bedingungen, die der Komponist zu Lebzeiten gewählt hat, verhindern jedoch die Benutzung bestehender Open-Content-Lizenzen, da sie eine zu geringe Flexibilität in der Rechtefreigabe besitzen. Eine daher selbstformulierte Ad-hoc-Lizenzierung bietet aber bei weitem nicht die Rechtssicherheit der bestehenden Open-Content–Lizenzen. Eine Überarbeitung dieser könnte hier Komponisten wie auch Nachlassverwaltern Anreize bieten, bisher nicht zugängliche Werke als Open Content freizugeben.

Schlüsselwörter: Open Content · Archiv · Musik · GEMA

1 Maki Ishii

Maki Ishii gehört zu den bedeutendsten japanischen Komponisten zeitgenössischer Musik des 20. Jahrhunderts. In mehr als 50 Jahren komponierte er über 180 Werke

Werkart	Anzahl
Solowerke	33
Kammerwerke	74
Orchester-/Konzertwerke	51
Bühnenwerke	17
Arrangements	4
Andere Werke	2
Alle Werke	181

Tabelle 1: Anzahl der Werke von Maki Ishii nach Werkart[1]

und schuf in Zusammenarbeit mit vielen Orchestern, Ensembles und Solisten eine eigene musikalische Welt, die Elemente westlicher Musik mit östlicher, insbesondere japanischer traditioneller Musik verbindet. Daneben organisierte er Konzerte wie auch ganze Musikfestivals und gab den Zuhörern Einblicke in die traditionelle und zeitgenössische Musik des westlichen und östlichen Musikkreises.

Mit dem unerwarteten Tod Maki Ishiis im April 2003 entstand ein umfangreicher musikalischer Nachlass, bestehend aus Noten (in Manuskripten und Drucken), Dokumenten und Unterlagen sowie Foto-, Audio- und Videoaufnahmen. Sowohl Musiker und Orchester, die mit ihm zusammengearbeitet hatten, als auch andere Interessierte fragen seither nach Informationen, Noten und anderen Materialien, um die Musik Maki Ishiis aufführen zu können. Dies setzt in gewisser Weise die Situation zu Lebzeiten Maki Ishiis fort, die durch die intensive persönliche Kommunikation zwischen Komponisten und Aufführenden gekennzeichnet war Es ist daher wichtig, den Zugang zu den Materialien des Nachlasses als eine Art Ersatz für diese Kommunikation so weit wie möglich zu erleichtern, um das Schaffen des Komponisten auch nach seinem Tode weiterleben zu lassen.

Zu diesem Zwecke habe ich als Sohn von Maki Ishii die Idee eines *Internetarchivs* aufgegriffen, wie es bereits vom Komponisten angedacht und im Ansatz realisiert worden war. Dieser Beitrag stellt die ersten Schritte auf dem Weg zu einem Internetarchiv des musikalischen Nachlasses Maki Ishiis dar. Dieses dient dazu, die durch den Tod des Komponisten abgebrochene Kommunikation mit den Musikern so gut wie möglich zu ersetzen und damit die Auseinandersetzung mit dem Kompositionswerk zu fördern.

Zentraler Aspekt des Archivs bei dem Aufbau ist die Zugrundelegung des Open–Content-Gedankens: Alle Inhalte – und hierunter insbesondere die bisher nicht pu-

1 Erklärung der Werkarten: *Solowerk* – für ein Soloinstrument; *Kammerwerk* – für mehrere Instrumente; *Konzertwerk* – für Orchester und ein oder mehrere Soloinstrumente; *Arrangement* – Bearbeitung des Werkes eines anderen Komponisten (oder traditionellen Werkes); *Anderes* – Film- bzw. Videomusik.

blizierten Noten – sind frei und kostenlos zugänglich. Die zunächst ins Auge gefasste Anwendung der besonders im Open-Content-Bereich populären Creative-Commons–Lizenzen kollidierte jedoch mit den bestehenden rechtlich-ökonomischen Rahmenbedingungen der Werke, so dass stattdessen auf eine Ad-hoc-Lösung zurückgegriffen werden musste, die möglicherweise eine geringere Rechtssicherheit bietet.

2 Der Nachlass

Die zahlreichen Anfragen zum Werk nach dem Ableben Maki Ishiis belegten die Dringlichkeit, den Nachlass möglichst schnell zu sichten und zugänglich zu machen. Dazu war es notwendig, sich zunächst einmal eine Übersicht zu verschaffen. Demnach umfasst der Nachlass insgesamt 181 Kompositionswerke (Tabelle 1 gibt eine Übersicht nach Werkart). Zu jedem Werk enthält er ein handschriftliches Originalmanuskript. Ist das Werk in einem Notenverlag publiziert worden, so sind dazu Belegexemplare der Notendrucke vorhanden.[2]

Sind die Noten in einem Verlag publiziert, so ist der Zugang für Musiker gesichert. Dies ist aber nur für die Hälfte der Werke der Fall, so dass für die anderen Werke die Notenmanuskripte die einzige Möglichkeit darstellen, das Werk aufzuführen. Entsprechend ist es ein vorrangiges Ziel des Internetarchivs, diese Notenmanuskripte zugänglich zu machen (Abschnitt 3.2).

Neben dem Notenmaterial umfasst der Nachlass eine große Anzahl von Dokumenten und Organisationsunterlagen zu Konzerten, Musikfestivals und anderen Musikprojekten von Maki Ishii, so etwa Programmhefte, Prospekte und Plakate, Zeitungsrezensionen und Zeitschriftenartikel, interne Projektkorrespondenz usw. Diese Materialien liegen überwiegend in Papierform vor, zu einem geringen Teil auch in digitaler Form. Neben den Noten stellte die erste Auswertung dieser Materialien die Grundlage für die Daten dar, die im Werkverzeichnis zusammengefasst wurden (Abschnitt 3.1). In weiteren Schritten sollen hieraus auch weitergehende Informationen zu Leben und Werk des Komponisten extrahiert werden.

Darüber hinaus enthält der Nachlass diverse Bild-, Audio- und Videoaufzeichnungen, etwa von Konzertproben, Aufführungen oder Radio- und Fernsehaufzeichnungen auf verschiedenen Arten von Datenträgern. Auch diese Materialien sind für ein Archiv zeitgenössischer Musik von großer Bedeutung, da sie den Musikern wertvolle Hinweise für die Interpretation und Aufführungspraxis der Werke liefern.

2 Darüber hinaus gibt es zu vielen Werken Kopien von Notenmanuskript bzw. Drucknoten, die der Komponist mit handschriftlichen Anmerkungen und Aufführungshinweisen versehen hat. Auch sie stellen eine wertvolle Quelle für die Aufführungspraxis dar und sollten daher in geeigneter Weise bereitgestellt werden.

Open Access

3 Das Open-Content-Archiv

Die Entscheidung darüber, welche Inhalte als Erstes in ein Internetarchiv eingebracht werden sollten, orientierte sich daran, welche Materialien und Informationen für potenzielle Aufführende der Musik Maki Ishiis am wichtigsten sein könnten. Nach Gesprächen mit Freunden meines Vaters und durch die Analyse der Anfragen von an den Werken Interessierten wurde entschieden, welche der Informationen vorrangig zu erschließen und veröffentlichen waren:

– Die häufigsten Anfragen betrafen Einzelinformationen zu den Werken; entsprechend wurde zunächst ein *Werkverzeichnis* mit Daten zu jedem Werk erstellt und in das Internetarchiv gestellt. Wichtig war es hier, eine für Musiker nützliche Auswahl an Daten zum Werk zu treffen, welche die Kommunikation mit dem Komponisten zumindest in Ansätzen ersetzen könnte.
– Der nächste Schritt befasste sich mit der Zugänglichmachung der nichtpublizierten Notenmanuskripte. Hier wurde ein Open-Content-Ansatz gewählt, wie er bereits von Maki Ishii angedacht worden war: Die Notenmanuskripte können als Dateien kostenlos heruntergeladen werden und sind so weltweit allen Musikern zugänglich. Als größte Schwierigkeit erwies sich die Wahl der Lizenz, da die Nutzungsrechte am *Notenbild* von denen am *Werk* unterschieden werden mussten.

Zukünftige Schritte werden die sukzessive Sichtung und Entscheidung über die Einbringung in das Internetarchiv umfassen. Es ist abzusehen, dass auch hier, wie bei den Noten, die urheberrechtliche Klärung und Lizenzierung einen nicht unerheblichen Einfluss darauf haben wird, inwieweit die Dokumente, Bild-, Audio- und Videoaufnahmen als *Open Content* zugänglich gemacht werden können.

3.1 Das Verzeichnis aller Werke

Für die Erstellung eines Werkverzeichnisses wurde eine Auswahl der Daten getroffen, die von den Musikern am häufigsten nachgefragt wurden. Die meisten Anfragen betreffen Informationen zu den Kompositionen; allerdings ist die Art der Fragen sehr unterschiedlich. So tragen Werke mal einen sprachunabhängigen Titel, mal mehrere übersetzte Titel (japanisch, deutsch, englisch).[3] Bei anderen Werken existieren zwei und mehr – für verschiedene Instrumente komponierte – Versionen,[4] die leicht durcheinandergebracht werden können. Auch ähnlich klingende japanische Titel führen leicht zu Verwechslungen zwischen den Werken.[5]

3 Zum Beispiel „Neun kleine Sätze", „Nine small movements", „Kokonotsu no shōhin".

4 Das Paradebeispiel hier ist das Werk „Lost Sounds" (englisch, deutsch) = „Ushinawareta hibiki" (japanisch), welches in sechs Varianten existiert: „Lost Sounds I" in Versionen A, B, C; „Lost Sounds II" in zwei im Titel nicht unterschiedenen Versionen und „Lost Sounds III".

5 Etwa „hibiki ranbu" – „hyōbyō no hibiki" – „rōrō no hibiki" – „yoru no hibiki" usw.

Andere Anfragen betreffen die Suche nach Werken mit einer bestimmten Instrumentalbesetzung (z. B. Klaviersolo, Streichquartett) oder den Zugang zu Noten oder Audioaufnahmen. Es wurden auch immer wieder Daten nachgefragt, deren Relevanz nicht unbedingt offensichtlich ist. Ein Beispiel ist die Frage nach Ort, Datum und Musikern der Erstaufführung. Zum einen dient dies wohl zur zeitlichen Einordnung des Werkes. Zum anderen werden die Namen der Erstaufführenden dazu verwendet, diese zwecks Fragen nach Interpretations- und Aufführungshinweisen sowie die Instrumentierung zu kontaktieren.

Die Beantwortung der Anfragen war vor Erstellung des Werkverzeichnisses mitunter arbeitsintensiv, da sie zu ausgedehnter Suche im weitgehend ungeordneten Nachlass führte. Schon aus diesem Grund wurde als erster Schritt ein fast vollständiges Verzeichnis aller Werke mit den Daten, die nachgefragt wurden, erstellt. Es umfasst für jedes Werk den Haupt- und Untertitel in Deutsch, Englisch und Japanisch; Entstehungs- und Erstaufführungsjahr; Musikart; Spieldauer; Instrumentalbesetzung; sowie Kontaktdaten für den Notenbezug.

Für das Internetarchiv wurde das Werkverzeichnis in drei Sprachen (deutsch, englisch, japanisch) übersetzt. Es ist so angelegt, dass die weiteren Materialien aus dem Nachlass (Notenmanuskripte, Diskographie, Bild-, Audio- und Videoaufzeichnungen) sukzessive eingebracht werden können, um den Musikern und anderen Interessierten eine zentrale Anlaufstelle für alle Informationen zu den Werken Maki Ishiis anzubieten.

3.2 Zugang zu den nichtpublizierten Noten

Die Aufführung eines Werkes ist das *sine qua non* der Musik: Ohne sie kann ein Werk nicht rezipiert werden. Eines der vorrangigen Ziele der Verwaltung des musikalischen Nachlasses ist es daher, die Noten der Werke den Musikern möglichst einfach zugänglich zu machen. Im Unterschied zum Werkverzeichnis spielen jedoch hier auch die rechtlich-ökonomischen Rahmenbedingungen eine große Rolle. Soll man die Noten über einen Verlag kommerziell vertreiben oder aber frei (also als Open Content) verbreiten? Ist rechtlich überhaupt die freie Weitergabe der Noten möglich? Und wenn ja, welcher lizenzrechtliche Rahmen sollte dafür gewählt werden? Ein Überblick über die rechtlich-ökonomische Situation zeitgenössischer Komponisten soll die Entscheidung, die Noten als Open Content freizugeben, und die sich dabei ergebenden Schwierigkeiten mit den gängigen Creative-Commons-Lizenzen erklären helfen.

Die ökonomischen Bedingungen des Bereichs der zeitgenössischen Musik unterscheiden sich wohl von denen der populären Musik. Es gibt kaum einen zeitgenössischen Komponisten, der seinen Lebensunterhalt ausschließlich aus den Einnahmen der Kompositionstätigkeit bestreiten kann. Regelmäßig dienen als „zweites Standbein" eine Lehrtätigkeit, die Arbeit als Instrumentalmusiker usw. Im Falle meines Vaters war dies die Durchführung zeitgenössischer Konzerte und Festivals in Japan, Europa und Amerika.

Für die Kompositionstätigkeit sind die Haupteinnahmequellen zum einen die Tantiemen aus den Urhebernutzungsrechten – den Gebühren aus der öffentlichen Aufführung in Konzerten, Abspielungen in Fernsehen und Radio und Urheberrechtsabgaben aus Tonträgerverkäufen; diese werden in der Regel über eine Verwertungsgesellschaft wie der deutschen GEMA verwaltet. Die Einnahmen aus dem Verkauf oder Verleih der Noten durch Notenverlage bilden eine zweite, aber weitaus geringere Einnahmequelle.

GEMA

Maki Ishii war schon früh Mitglied der Gesellschaft für Musikalische Aufführungs- und Vervielfältigungsrechte (GEMA) geworden, die gegen Übertragung der Urhebernutzungsrechte der Werke die Gebühren aus diesen Rechten weltweit einzieht und als Tantiemen an die Mitglieder ausschüttet.

Die GEMA ist eine privatrechtlich organisierte Gesellschaft, die in Deutschland die kollektive Wahrnehmung und Durchsetzung der Urheberrechte ihrer Mitglieder organisiert.[6] Der Künstler schließt dazu einen so genannten „Berechtigungsvertrag" ab, mit dem die Nutzungsrechte an *allen* Werken *ausschließlich* an die GEMA übertragen werden.[7] Sie tritt als Treuhänderin dieser Rechte auf, indem sie diese wahrnimmt und durchsetzt. Die Einnahmen werden regelmäßig nach einem in der Satzung festgeschriebenen Schlüssel als Tantiemen an die Mitglieder ausgeschüttet.

Für den Künstler liegen die Vorteile darin, dass die GEMA die weltweite Rechtedurchsetzung[8] und den Gebühreneinzug organisiert, was in dieser Weise für den einzelnen Künstler unmöglich wäre. Für meinen Vater – und sicherlich auch für andere Komponisten seiner Generation – wog dieser institutionelle Vorteil den eher hypothetischen Nachteil auf, die Verfügungsmacht über alle Nutzungsrechte zugunsten der GEMA aufgeben zu müssen. Dies umso mehr, da das Internet als Kommunikations- und Verbreitungsmedium, welches alternative „Geschäftsmodelle" möglich macht, noch gar nicht in der heute bekannten Form existierte.

Für den Kontext der Notenverbreitung sieht der Vertrag mit der GEMA jedoch eine Ausnahme[9] vor: Das so genannte *graphische Recht*, also das *Recht am Notenbild* wird explizit von der Übergabe der Nutzungsrechte ausgenommen, da dieses Recht von den Notenverlagen wahrgenommen wird. Im Gegensatz zu den Nutzungsrechten kann

6 Die Angaben dieses Abschnittes stammen aus Kreile et al. (2006).

7 Ab Beginn der Mitgliedschaft; dies umfasst auch alle zum Zeitpunkt des Vertragsschlusses *zukünftigen* Werke.

8 Dies geschieht durch Kooperation zwischen den jeweils national agierenden Verwertungsgesellschaften. Ein Musiker in Japan etwa führt die Gebühren an die dortige Verwertungsgesellschaft (*JASRAC*) ab, die diese dann an die GEMA abführt, um sie ihrem Mitglied gutzuschreiben. Dies gilt dann auch z. B. für einen deutschen Musiker und ein Mitglied der *JASRAC* usw.

9 Es gibt mindestens eine weitere Ausnahme (das sog. „große Recht"), welches jedoch für das vorliegende Thema nicht von Bedeutung ist.

	publiziert	nicht publiziert	gesamt	Anteil 'publiziert' in %
1950–1959	0	2	2	0
1960–1969	13	6	19	68
1970–1979	25	11	36	69
1980–1989	30	36	66	45
1990–1999	10	29	39	26
2000–2003	4	13	17	24
ohne Jahr	0	2	2	0
Summe	82	99	181	45

Tabelle 2: Anzahl verlagspublizierter und nicht publizierter Werke per Schaffensdekade

damit das GEMA-Mitglied frei über das Recht am Notenbild seiner Werke verfügen, solange es dieses nicht einem Verlag übergeben hat.

Verlagspublikation

Generell ist es das wichtigste Ziel für Komponisten zeitgenössischer Musik – und sicherlich auch denen anderer Musiksparten –, dass man sich mit ihrer Musik auseinandersetzt, dass sie zur Aufführung gebracht und dadurch gehört wird. Im Unterschied zu anderen Künsten ist das Kompositionsprodukt, die Noten, nicht das Kunstwerk an sich. Erst durch die Vermittlung des Musikers kann das Werk rezipiert werden. Damit gehört der möglichst einfache Zugang interessierter Musiker zu den Noten zu einem wichtigen Ziel sowohl für Komponisten als auch für diejenigen, welche deren Nachlass verwalten.

Wie verfuhr Maki Ishii mit dem „Recht am Notenbild"? Man könnte annehmen, dass er alle oder die meisten Werke in einem Verlag publiziert hat, um eine möglichst große Verbreitung zu erreichen und Erlöse aus dem Verkauf bzw. Verleih zu erhalten. Die Sichtung des Nachlasses ergab jedoch, dass der Komponist weniger als die Hälfte seiner Werke in einem Verlag publiziert hatte (Tabelle 2).

Hinzu kommt, dass der Anteil der publizierten an den neu geschaffenen Werken im Laufe der Zeit stetig zurückging. Während Maki Ishii in den 1960er und 1970er Jahren noch über zwei Drittel seiner neuen Werke publizieren ließ, war dieser Anteil in den letzten dreizehn Jahren auf ein Viertel zurückgegangen.

Ein Indiz für diesen Rückgang kann man aus den Daten der Verkäufe der publizierten Noten herauslesen. Demnach erzielten in einem Verlag[10] bei insgesamt 33 Verkaufsnoten in den letzten Jahren lediglich fünf Noten zusammen zwischen 74 % und 82 % des Ertrags. Nur wenige der publizierten Noten konnten für Verlag und

10 In diesem Verlag sind über die Hälfte der publizierten Werke Maki Ishiis verlegt.

Komponisten einen nennenswerten Ertrag generieren. Zudem machte (und macht) der Ertrag aus den Publikationen nur einen Bruchteil der GEMA-Tantiemen aus. Wenn man zusätzlich in Betracht zieht, dass für eine Publikation kostenintensiv eine Druckvorlage aus dem handschriftlichen Manuskript erzeugt werden muss, so lohnt sich für nur wenige Werke eine Verlagspublikation. Es ist den Verlagen hoch anzurechnen, dass sie der Entscheidung über die Verlegung von Noten zeitgenössischer Musik nie ein rein kommerzielles Kalkül zugrunde gelegt haben.

Gerade vor dem Hintergrund der zunehmenden Verfügbarkeit technischer Mittel zur Verbreitung und Kommunikation – von Kopiergeräten bis hin zum Internet – schwindet die Notwendigkeit, die Noten über Verlage verbreiten zu müssen. Dagegen bieten die neuen Mittel den Vorteil der Intensivierung der direkten Kommunikation des Komponisten mit den Musikern.

Wie bereits erwähnt, beantwortete Maki Ishii die Anfragen von Musikern aus aller Welt stets selbst. Im Falle von nichtpublizierten Noten fertigte er Kopien des Notenmanuskriptes an und versendete sie an den Musiker, ohne dass Letzterem die Kosten in Rechnung gestellt wurden. Es konnte allerdings auch vorkommen, dass er stattdessen ein neues Werk schuf. Der für ihn entscheidende Aspekt war der ständige Kontakt und Austausch mit den Aufführenden. Dies war sowohl vom künstlerischen Aspekt her wichtig, konnte aber auch die Anbahnung von neuen Projekten oder Kompositionsaufträgen bedeuten.

Open-Content-Zugang zu den Noten des Nachlasses

Wie sollte nun, nach dem Tod des Komponisten, mit dem Zugang zu den nichtpublizierten Noten verfahren werden? Wie den bisherigen Ausführungen entnommen werden kann, stellt die Verlagspublikation für die meisten Werke keine optimale Lösung dar. Einzig bei den Orchester-, Konzert- und Bühnenwerken mit umfangreichem Notenmaterial[11] ist ein kommerziell organisierter Notenverleih anderen Möglichkeiten (Einzelkopien versenden bzw. im Internet zum Download anbieten) vorzuziehen.

Allerdings konnte auch die Praxis meines Vaters, Kopien der Notenmanuskripte herzustellen und zu versenden, nicht mehr aufrechterhalten werden. Der Mehrwert, den der Komponist aus der Kommunikation mit Aufführenden zog, besteht für die Nachlassverwalter in dieser Form nicht mehr, so dass die Kosten den Nutzen nicht aufwiegen.

Das prinzipielle Problem des Zugangs zu den Noten scheint im Bereich der zeitgenössischen Musik verbreitet zu sein: In einem persönlichen Gespräch schilderte ein Musiker, dass viele der Werke nach der Erstaufführung nur noch schwer zugänglich seien. Die meisten Werke würden nicht publiziert, und gerade nach dem Tod des Künstlers sei der Zugang in der Regel so mühsam, dass meist die Mühen gescheut und deshalb viele interessante Werke nicht aufgeführt würden. Entsprechend äußerten

11 Partitur und Stimmauszüge können hier einige hundert Seiten umfassen.

auch viele Freunde meines Vaters ihre Sorge über die zukünftige Zugänglichkeit der Noten.

All diese Faktoren zusammengenommen – von der ökonomischen Situation über das Kosten-Nutzen-Verhältnis von Verlagspublikation und Kopieren der Noten bis hin zur Sorge der Freunde über den Zugang zu den Noten – bewogen mich dazu, die nichtpublizierten Notenmanuskripte zu scannen und als Open Content in das Internetarchiv zu stellen.

Die Idee dazu war nicht neu: Maki Ishii hatte mithilfe eines Freundes[12] eine eigene „offizielle" Website einrichten lassen, für die er kontinuierlich Inhalte schuf. Mit dem Ziel, den Musikern die nichtpublizierten Werke einfacher als bisher zukommen zu lassen, setzte er 2001 das Notenmanuskript eines seiner Werke zum kostenlosen Download auf die Website. Der kostenlose Bezug der Noten sollte die Auseinandersetzung mit dem Werk fördern. Gebühren (der Verwertungsgesellschaft) würden dann erst bei einer öffentlichen Aufführung oder dem Verkauf von Aufnahmen anfallen.

Aus all diesen gegebenen Gründen fiel die Entscheidung auf die Verbreitung der nichtpublizierten Noten als Open Content über das Internetarchiv Maki Ishiis.

Bearbeitung der Originalmanuskripte

Von den etwa einhundert nichtpublizierten Werken sind knapp die Hälfte Orchester-, Konzert- und Bühnenwerke, deren Umfang eine Internetpublikation schwierig gestaltet. Es wurden daher die etwas mehr als 50 Solo- und Kammerwerke für die erste Internetpublikation ausgewählt. Diese umfassen knapp eintausend DIN-A3-Seiten, die im Laufe der letzten zwei Jahre eingescannt wurden. Anschließend wurden die Seiten bearbeitet und die Dateigröße so reduziert, dass ein siebenseitiges Werk einer etwa drei Megabyte großen Datei entspricht; alle für Musiker wichtigen Details bleiben so bei Ausdruck auf DIN-A3-Papier erhalten. Deck- und Informationsseite vervollständigen die Internetpublikation. Abbildung 1 zeigt einen Ausschnitt aus einem für die Internetpublikation aufbereiteten Notenmanuskript.

Lizenzierung – Creative Commons?

Vor der Veröffentlichung der Notenmanuskripte gab es noch eine wichtige Frage zu klären: Unter welchen Lizenzbedingungen sollten sie veröffentlicht werden? Da die Noten kostenlos zum Download angeboten werden, geht es hier weniger um den Schutz des Werkes als vielmehr um die Rechtssicherheit der Musiker als Nutzer. Bei dem Open-Content-Angebot der Noten könnte man durchaus annehmen, dass auch das Werk selbst frei genutzt (z. B. öffentlich aufgeführt) werden kann. Da aber die Werknutzungsrechte, wie oben ausgeführt, nicht bei den Rechtsnachfolgern von Maki

12 Ich danke Herrn Yūsuke Watanabe, der die erste offizielle Website für meinen Vater erstellte und betreibt, für die Mitteilung der Beweggründe Maki Ishiis betreffend der Website und der Internetpublikation eines Werkes.

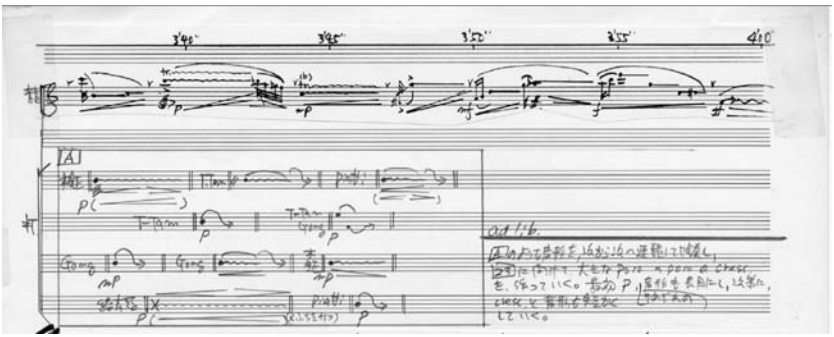



Abbildung 1: Notendetail aus Maki Ishii „Ko-ku" für Ryuteki und Percussion, op. 74 (1987)

Ishii, sondern bei der GEMA liegen, können Erstere die Nutzungsrechte gar nicht freigeben. Es ist also sehr wichtig, den Umfang der Freigabe der Noten explizit zu machen.

Zunächst war geplant, die Notenmanuskripte unter eine der im Open-Content–Bereich wohl am häufigsten eingesetzten Creative-Commons-Lizenzen zu setzen. Das Creative-Commons-Projekt[13] hat es sich zur Aufgabe gemacht, innerhalb des geltenden Urheberrechts flexible Lizenzen anzubieten, so dass das Recht einer „freien Kommunikation und Zusammenarbeit nicht im Wege steht".[14] Ein großer Vorteil beim Einsatz einer dieser Lizenzen besteht darin, durch Anwendung einer standardisierten Lizenz eine erhöhte Rechtssicherheit gegenüber individuell oder mit teurem Rechtsbeistand gestalteten Lizenzen zu erhalten.

Es stellte sich jedoch heraus, dass die Creative-Commons-Lizenzen für den Fall von freien Notenmanuskripten bei rechtlich gebundenen Werken nicht anwendbar sind. Die Ursache liegt darin, dass diese Lizenzen keine Unterscheidung zwischen dem Werk und dem Notenbild machen und stattdessen *pauschal* die Rechte zur Vervielfältigung, Verbreitung und öffentlichen Aufführung freigegeben werden:

> „Sie dürfen: den Inhalt vervielfältigen, verbreiten und öffentlich aufführen."[15]

Die FAQ der deutschen Creative-Commons-Organisation führen dementsprechend aus:

Frage: „Was muss ich beachten, wenn ich Mitglied der GEMA bin und eine Creative–Commons-Lizenz benutzen will?"

13 http://creativecommons.org/

14 Aus der FAQ der *Creative Commons Deutschland*, http://de.creativecommons.org/faq.html#welchesproblem_anwort.

15 http://creativecommons.org/licenses/by-nc-nd/2.5/deed.de

Antwort: „[. . .] Durch den Wahrnehmungsvertrag mit der GEMA überträgt der Künstler eine Reihe ausschließlicher Nutzungsrechte an all seinen musikalischen Schöpfungen. Die Nutzung einer Creative-Commons-Lizenz für GEMA-Mitglieder ist somit zurzeit nicht möglich, da diese nicht mehr über die hierfür erforderlichen Rechte verfügen."[16]

Für die Werkarten, auf die sich die *Creative Commons* in erster Linie bezieht,[17] und für ihre Zielgruppen – etwa junge Künstler, die noch keine rechtliche Bindung zu Verwertungsgesellschaften haben – mag die pauschale Freigabe der Rechte am Werk vorteilhaft sein. Für den hier beschriebenen Fall ist eine solche Rechtefreigabe gar nicht möglich.

Für die Zielgruppe „etablierter" Künstler, deren bestehende rechtlich-ökonomische Situation eine pauschale Freigabe der Rechte nicht zulässt, müssten angepasste Creative-Commons-Lizenzen entworfen werden. Es ist gut vorstellbar, dass (nicht nur) zeitgenössische Komponisten von einem solchen Angebot regen Gebrauch machen würden, was wiederum die Idee des Open Content fördern würde.

Die (vorläufige) Notenlizenz

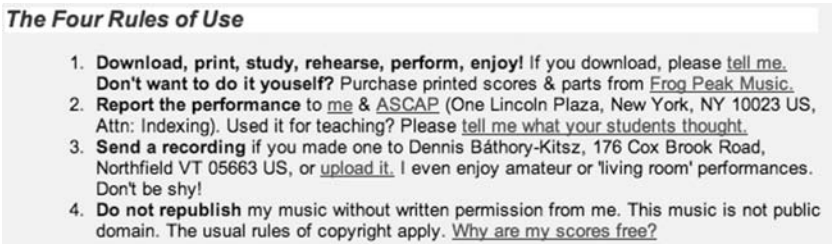
Da die derzeitigen Creative-Commons-Lizenzen nicht für die Notenveröffentlichung geeignet sind, musste eine andere Lösung gefunden werden. Als Vorgehensweise wurde dazu die Suche nach Beispielen anderer Komponisten, die Noten im Internet veröffentlicht haben, gewählt; diese dienten als Vorbild für einen eigenen Ad-hoc–Lizenztext. Dieser Vorgehensweise fehlt sicherlich die Eleganz und Rechtssicherheit einer Creative-Commons-Lizenz, sie sollte aber zunächst den oben genannten Anforderungen genügen, die Musiker darüber zu informieren, dass die Nutzung der Noten den Bedingungen der GEMA unterliegt.

Die Idee, Musiknoten urheberrechtsgeschützter Werke im Internet zum freien Download anzubieten, ist nicht neu. Eine kursorische Suche ergab nicht überraschend, dass eine Reihe von Komponisten ihre Noten zum freien Download anbietet. Die Lizenzhinweise zeigen auch, dass die Komponisten häufig Mitglied einer Verwertungsgesellschaft sind, also die Werknutzungsrechte nicht freigeben können.

Die Form der Kennzeichnung gleicht sich bei den gefundenen Beispielen: Praktisch alle geben den von Noten von Verlagen her bekannten Hinweis mit Copyrightzeichen, Jahr und Namen des Rechteinhabers (Verlag oder Komponist) an, z. B. „© 2006 Maki Ishii". Einige Beispiele fügen auch die Einschränkung der Rechte hinzu, etwa „All Rights Reserved" oder auch „The rights of this music belong to the publisher (composer)".

16 http://de.creativecommons.org/faq.html#was_mus_ich_antwort

17 Als „Werkformat" werden bei der Wahl einer Lizenz neben „Sonstiges" folgende Formate angeboten: Audio, Video, Bild, Text, Interaktiv (http://creativecommons.org/license/index_html?lang=de).


The Four Rules of Use

1. **Download, print, study, rehearse, perform, enjoy!** If you download, please tell me. **Don't want to do it youself?** Purchase printed scores & parts from Frog Peak Music.
2. **Report the performance** to me & ASCAP (One Lincoln Plaza, New York, NY 10023 US, Attn: Indexing). Used it for teaching? Please tell me what your students thought.
3. **Send a recording** if you made one to Dennis Báthory-Kitsz, 176 Cox Brook Road, Northfield VT 05663 US, or upload it. I even enjoy amateur or 'living room' performances. Don't be shy!
4. **Do not republish** my music without written permission from me. This music is not public domain. The usual rules of copyright apply. Why are my scores free?

Abbildung 2: „Four Rules of Use" (Dennis Báthory-Kitsz)

Ein derartiger Hinweis alleine erschien jedoch nicht ausreichend, um Musiker über die Nutzungsbeschränkungen zu informieren. Der bei der Suche entdeckte Lizenztext eines (mir unbekannten) Komponisten dagegen erfüllte diese Bedingung; man kann davon ausgehen, dass diesen Komponisten ähnliche Gründe wie die hier vorgestellten dazu bewogen, seine Noten im Internet zu verbreiten.

Der Kurzhinweis in den zum Download angebotenen Noten des Komponisten[18] entspricht der oben genannten Form: „Copyright © 2006 by Dennis Báthory-Kitsz (ASCAP). All rights reserved."[19] Interessanter dagegen sind seine mit *Four Rules of Use*[20] überschriebenen detailliert formulierten Lizenzbedingungen (Abbildung 2).

Die auch für das Open-Content-Konzept der Noten Maki Ishiis wichtigen Elemente, die sich in diesen *Four Rules* widerspiegeln, sind:

Auseinandersetzung mit dem Werk Der wichtigste Aspekt für den Komponisten, die Auseinandersetzung mit seinen Werken, wird hier an den Anfang gesetzt: „Download, print, study, rehearse, perform, enjoy".

Kommunikation zwischen Komponisten und Aufführenden In allen Regeln wird der Wunsch nach Kommunikation mit den Nutzern explizit ausgedrückt, sei es bei einem Download (Regel 1), einer Aufführung bzw. Nutzung in der Lehre (Regel 2), bei Aufnahmen (Regel 3) und indirekt durch die Freigabe der Noten.[21]

Rechtliche Schranken Regeln 2 und 4 schließlich weisen auf die rechtliche Einschränkung der freien Nutzung der Musik und der Noten hin. In Regel 2 wird die

18 Es handelt sich dabei um Dennis Báthory-Kitsz, siehe http://maltedmedia.com/people/bathory/cat-index.html.

19 Die *ASCAP (American Society of Composers, Authors, and Publishers)* ist das US-Äquivalent zur deutschen GEMA.

20 Siehe z. B. http://maltedmedia.com/people/bathory/cat-solo.html. Die Seiten für andere Werkarten enthalten gleichlautende Regeln.

21 Der Link „Why are my scores free?" (http://maltedmedia.com/people/bathory/cat-free.html) führt zu einer lesenswerten Begründung für die Freigabe der Noten, die sich teilweise mit den in diesem Beitrag genannten Argumenten deckt.

Meldepflicht an die Verwertungsgesellschaft – und damit implizit auch die Gebührenpflicht – im Falle von öffentlichen Aufführungen genannt.[22] Regel 4 verbietet für die Noten die (kommerzielle) Wiederveröffentlichung ohne vorherige Genehmigung.

Damit vereinen die *Four Rules* die für den Komponisten wichtigsten Elemente: Auseinandersetzung mit seiner Musik, Kommunikation mit den Musikern und die rechtlichen Beschränkungen bei der Nutzung durch die Verwertungsgesellschaften und bei der Weitergabe der Noten.

Zum Zeitpunkt der Erstellung dieses Beitrags war der endgültige Lizenztext für die Veröffentlichung der Notenmanuskripte Maki Ishiis noch nicht festgelegt; er entspricht aber in Geist und Form den vorgestellten *Four Rules of Use*. Der fertige Lizenztext wird auf einem speziellen Informationsblatt in den Hauptsprachen (japanisch, englisch, deutsch) den Noten vorangestellt werden, damit die Musiker direkt über die Nutzungsbedingungen der Noten und des Werkes informiert werden.

4 Die nächsten Schritte

Neben den Informationen zu den Werken sowie den Notenmanuskripten enthält der Nachlass noch eine Reihe von weiteren Daten, die aufbereitet für das Internetarchiv von Bedeutung sind. Zusätzlich zum kontinuierlichen Ausbau der bestehenden Informationen und Daten sind als nächste Schritte eine Diskographie, Beschreibungen von ungewöhnlichen Instrumenten, die Maki Ishii in seinen Kompositionen eingesetzt hat, sowie die Erschließung der Foto-, Audio- und Videomaterialien geplant. Insbesondere letztere Materialien könnten sich in rechtlicher Hinsicht als schwierig erweisen, da die Urheberlage einiger dieser Materialien nicht geklärt ist. Auch der Arbeits- und Kostenaufwand der Digitalisierung ist nicht zu unterschätzen: Speziell ältere Materialien (z. B. Tonbänder aus den 1960er Jahren) bedürfen einer vorsichtigen Handhabung.

Die Verfolgung des Open-Content-Prinzips für Materialien zeitgenössischer Musik ist aus meiner Sicht auch angesichts der beschriebenen Herausforderungen ein lohnenswertes Ziel. Trotz der rechtlichen Beschränkungen – vorrangig durch die Übertragung der Nutzungsrechte an eine Verwertungsgesellschaft – gibt es genügend Materialien, deren Open-Content-Verbreitung (nicht nur) bei Musikern auf Interesse stoßen könnte, ohne dabei den Intentionen und Zielen des Künstlers entgegenzulaufen. Das hier gezeigte Beispiel der nichtpublizierten Notenmanuskripte zeigt dies deutlich, wobei es unwesentlich ist, dass es sich hier um Noten aus einem Nachlass handelt.

22 Nach meinem Dafürhalten ist diese Regel nicht sehr klar formuliert: Während die Meldung einer Aufführung an den Komponisten („report to me“) eher als Bitte anzusehen ist, ist die Meldung an die *ASCAP* („report to *ASCAP*“) urheberrechtlich vorgeschrieben.

Ein interessantes Ergebnis ist auch die Nichtanwendbarkeit der Creative-Commons-Lizenzen. Ihr Anspruch, möglichst viele der Nutzungsrechte freizugeben, kollidiert mit der bestehenden rechtlich-ökonomischen Situation des Komponisten. Das Modell der Verwertungsgesellschaften, im Gegenzug zur Übergabe der Nutzungsrechte ihre weltweite Durchsetzung zu organisieren, bietet sicherlich nach wie vor für viele Künstler einen hohen individuellen Anreiz. Es wäre zu wünschen, dass sich diese gegensätzlichen Modelle einander annäherten – ein erster Schritt könnte die Formulierung einer Creative-Commons-Lizenz für die Open-Content-Verbreitung von Noten (und anderen Materialien) sein. Aber auch die Verwertungsgesellschaften müssen sich überlegen, ob ihr Ausschließlichkeitsmodell unter den heutigen technischen Kommunikations- und Interaktionsbedingungen noch adäquat ist.

Literatur

Kreile, R., Becker, J. und Riesenhuber, K. (2006), *Recht und Praxis der GEMA*, de Gruyter, Berlin.

Die neuen Ossis*

JOSCHA MAY

(CC-Lizenz siehe Seite 563)

„Vati, ich möchte OSSI werden", verkündete Friedemann beim Frühstück. Seine Leidenschaft war das Programmieren. Das Abitur stand kurz bevor und er war froh, sich endlich entschieden zu haben, was er mit seinem Talent anfangen wollte. Open–Source-Software-Ingenieur war genau das Richtige. Es passte perfekt zu ihm.

„Na da hättest du aber ein bisschen früher aufstehen müssen. Der Zug ist doch längst abgefahren mein Junge", antwortete der Vater mit einem leichten Schmunzeln und wunderte sich, was seinen Sohn wohl zu so einer Aussage bewegte. Die Mauer war schließlich schon vor langer Zeit gefallen und im Allgemeinen waren alle ganz froh, dass der Spuk vorbei war.

„Ach iwo", sagte Friedemann. „Es geht doch gerade erst richtig los. Die Bewegung wird wachsen."

„Ich kann dich gut verstehen mein Junge", sagte der Vater, als er merkte, dass sein Sohn ernsthaft darüber reden wollte. „Ich habe auch mal so gedacht, aber glaub mir, das bringt nichts. Das System ist zum Scheitern verurteilt. Die Menschen arbeiten nun mal nicht aus Lust an der Freude."

„Doch natürlich, in einer starken Gemeinschaft, und wenn jeder das tut, was er am besten kann und was ihm am meisten liegt, kommt einem die Arbeit gar nicht wie Arbeit vor."

Der Vater rollte mit den Augen und sein Gesicht verzog sich etwas über die Verwirrungen seines Sohnes. Auf der einen Seite freute es ihn zwar, dass der Junge begann, sich für Politik zu interessieren und nicht immer nur Computer im Kopf hatte. Aber dann gleich so was! Sein Sohn ein Sozialist oder gar ein Kommunist?

„Das schlag dir mal schnell wieder aus dem Kopf", sagte er, diesmal etwas barscher. „Das ist reine Zeitverschwendung."

Aber Friedemann insistierte: „Doch Vati, es gibt derzeit viele erfolgreiche Projekte. Ich habe schon einige Teilnehmer kennen gelernt. Es sind sehr sympathische Menschen, die mit Begeisterung von ihrer Arbeit erzählen."

---

* Diese Kurzgeschichte aus unserem Schreibwettbewerb „Open Fiction" wurde vom Redaktionsteam mit dem ersten Preis ausgezeichnet.

Der Vater zuckte zusammen. War sein Filius etwa in die Fänge einer sozialistischen Sekte geraten? Sein Gehirn arbeitete jetzt auf Hochtouren. Sein Sohn war ihm in letzter Zeit tatsächlich etwas orientierungslos vorgekommen. Aber wie kam er plötzlich an diese Sekte bzw. wie kam diese Sekte an ihn? Er versuchte, sich seinen Schrecken nicht anmerken zu lassen. „Die würde ich auch gerne mal kennen lernen", sagte er schließlich.

Friedemann kannte seinen Vater ganz genau. Er sah sofort, dass er ihn noch lange nicht überzeugt hatte, und dass dieser alles andere als glücklich über seine Entscheidung war. Das musste daran liegen, dass sein Vater diese freie Art der Programmierung für eine brotlose Kunst hielt. Aber auch, wenn es nicht leicht war, konnte man mit OS-Programmierung durchaus gutes Geld verdienen oder sich zumindest einen Namen machen, der einem später zu einem guten Job verhelfen konnte. Geld war auch nicht sein wichtigstes Motiv. Ihn faszinierte die lockere und dennoch leidenschaftliche Arbeitsweise, die in Open-Source-Projekten üblich war.

Aber sein Vater setzte schon nach. Offenbar war ihm das mit dem Kennenlernen doch nicht so wichtig. „Wer mit zwanzig nicht an Marx glaubt, der hat kein Herz . . . ", sagte er und schaute, als wenn gleich etwas sehr Bedeutendes folgen würde: „. . . und wer mit dreißig noch an Marx glaubt, der hat kein Hirn."

Friedemann verstand jetzt nur noch Bahnhof. Er fürchtete, sein Vater würde ihm jetzt einen endlosen Vortrag halten. Er konnte mit diesem philosophischen Kram nicht so viel anfangen. Verstohlen warf er einen Blick auf die Uhr und stellte erleichtert fest, dass er los zum Unterricht musste. Er stand auf und nahm seinen Rucksack: „Ich muss jetzt los Papa. Lass uns drüber reden, wenn ich zurück bin. Und mach dir keine Sorgen, es gibt genügend bezahlte Open-Source-Ingenieure", sagte er beim Hinausgehen.

„Was hatte sein Sohn da gerade noch gesagt? Open-Source-Ingenieure? War das der Name dieser neokommunistischen Sekte?" Der Vater war zwar lange nicht so computerbegabt wie sein Sohn, aber er wusste sehr wohl, wie man mit *Google* umgeht. Er beschloss, genauer nachzuforschen, wo sein Sohn da hineingeraten war.

Nach einer halben Stunde Internetrecherche war ihm ein Stein vom Herzen gefallen. Er lachte über sich selbst und über dieses obskure Missverständnis. Sein Sohn ein OSSI. Das bedeutete heutzutage also *Open-Source-Software-Ingenieur* zu sein. Dass sein Sohn eine Art Wohltätigkeitsprogrammierer werden wollte, gefiel ihm zwar auch nicht, aber alles halb so schlimm wie eine kommunistische Sekte. Und hatte er nicht gesagt, es gäbe auch bezahlte Programmierer? Es schien ihm zwar nicht besonders logisch, dass jemand einen anderen dafür bezahlt, ein Programm zu schreiben, welches dann später verschenkt wird, aber schließlich gab es ja auch Werbegeschenke. Und die Leute, die Werbegeschenke entwickelten, wurden ja auch bezahlt. So ähnlich musste das wohl auch mit Open-Source-Software sein. Er beschloss, mit Friedemann darüber zu sprechen, wenn dieser aus der Schule zurück war.

„Wie hast du dir das eigentlich vorgestellt als OSSI. Wovon willst du leben?“, fragte er seinen Sohn beim Abendessen.

Friedemann kaute noch kurz zu Ende: „Ich würde gern zunächst meine Fähigkeiten verbessern, Projekterfahrung sammeln und mir in der Szene einen Namen machen.“

„Na toll, und irgendwann bist du dann ein bekannter Wohltätigkeitsprogrammierer. Und was dann? Klebst du dir ein rotes Kreuz auf die Stirn und nennst dich Mutter Friedemann? Oder worauf soll das hinauslaufen?“

Friedemann war den rauen, sarkastischen Umgangston seines Vaters gewöhnt und blieb ruhig: „Das ist heute nicht mehr ganz so, wie du dir das vorstellst Vater. Mit etwas Projekterfahrung habe ich gute Chancen, damit Geld zu verdienen und vielleicht sogar Projektmanager zu werden.“

„Warum wird jemand bezahlt, damit er kostenlose Softwareprogramme schreibt?“, fragte der Vater. Seine Skepsis war noch lange nicht verflogen.

„Weil es Organisationen und Unternehmen gibt, die ein Interesse daran haben, dass bestimmte Software existiert.“

„Du meinst so was wie Wohltätigkeitsorganisationen?“

„Nein, das können auch ganz normale Unternehmen wie *IBM* oder *SAP* sein, die nichts anderes als ihre Gewinne im Sinn haben. Die denken dabei nur an sich und handeln strategisch. Sie wollen einfach, dass bestimmte Software für jedermann verfügbar ist und sind nicht darauf angewiesen, Kopien davon zu verkaufen.“

„Das musst du mir aber genauer erklären.“ Der Vater war jetzt neugierig geworden.

Friedemann überlegte, wo er anfangen sollte: „Es kommt in der Industrie immer wieder zu so genannten Paradigmenwechseln.“

„Was meinst du damit?“

„Damit ist gemeint, dass sich die maximale Wertschöpfung auf andere Stufen innerhalb der Wertschöpfungskette verlagert.“

„Häh?“ Jetzt verstand der Vater nur noch Bahnhof.

„Ganz früher z. B. gab es so was wie Softwareindustrie eigentlich gar nicht.“

„Na da gab es ja auch noch keine Computer.“

„Nein, ich meine in den Anfängen der Computerindustrie. Da drehte sich alles um Hardware. Software wurde den Kunden meist dazugegeben, damit sie die Hardware benutzen konnten. Sie war als Produkt für sich nicht üblich. Es gab zwar auch Softwareunternehmen, aber die waren immer auf bestimmte Rechnerarchitekturen spezialisiert und eher so was wie Zulieferer für die Hardwareindustrie.“

„Ja und?“

„Und dann hat sich *IBM* für eine standardisierte PC-Architektur entschieden, die von jedem nachgebaut werden konnte, was *IBM* auch gestattete. So gelang es *IBM*, ihre Architektur als Standard zu etablieren, der sich weltweit durchsetzte und mit der Zeit alle anderen PC-Architekturen, wie z. B. den *Amiga*, verdrängte.“

Friedemann sah, dass sein Vater nicht den geringsten Schimmer hatte, worauf er hinauswollte. „Die Leute von *IBM* hatten die Konsequenzen ihrer Entscheidung

Open Access

nicht erkannt. Der Wettbewerb um Hardware verschärfte sich drastisch und die Margen wurden immer kleiner. Gleichzeitig entstanden auf der ganzen Welt Softwareunternehmen, die verschiedenste Programme für alle möglichen Anwendungsbereiche entwickelten. Die Anwender benutzten ja jetzt alle dieselbe Hardwarearchitektur. Das große Geld wurde fortan mit Software verdient anstatt mit Hardware und Bill Gates rieb sich die Hände. Das meinte ich mit Paradigmenwechsel und Veränderungen in der Wertschöpfungskette."

Der Bogen war noch nicht komplett gespannt und der Vater runzelte die Stirn: „O.K., und wo kommt jetzt Open-Source-Software ins Spiel? Das klingt ja erstmal nicht nach Geld verdienen?"

„Naja, z. B. gibt es Unternehmen, die ein Interesse daran haben, wieder so einen Paradigmenwechsel herbeizuführen. Sie wollen dies erreichen, indem sie Software oder zumindest bestimmte Software zum billigen Massenprodukt machen."

Dem Alten ging so langsam ein Licht auf. Friedemann führte weiter aus: „*IBM* ist wieder ein gutes Beispiel. Die haben nie ganz verkraftet, dass sie damals so viel Marktmacht an *Microsoft* verloren haben und sind inzwischen einer der Hauptförderer von *Linux*. Sie wollen Betriebssystemen durch Bereitstellung einer kostenlosen Variante ihren Wert nehmen und so ihre Hauptkonkurrenten *SUN* und *Microsoft* schwächen. Dadurch ergibt sich für *IBM* die Möglichkeit, die Wertschöpfung auf höhere Stufen, wie z. B. Sicherheits- und Managementsoftware zu verlagern."

„Und wenn die irgendwann auch nichts mehr wert sind?"

„Dann verdienen die Firmen ihr Geld mit Beratung oder mit speziellen Anpassungen an die Bedürfnisse der Kunden."

„Und du möchtest also später solche Open-Source-Projekte managen."

„Ja das wäre toll. Die Arbeitsweise in solchen Projekten ist ganz anders als in der normalen Softwareentwicklung. Es geht nicht darum, den Leuten Arbeitspakete zuzuweisen und ihnen Abgabetermine vorzuschreiben. Sie entwickeln von selbst die Software, die sie brauchen. Es geht eher darum, sie bei Laune zu halten. Die Arbeitsatmosphäre in solchen Projekten ist super angenehm, da ja fast alle freiwillig mitmachen."

„Das klingt ja nach Animateur im Ferienclub. Bist du dann der Spaßmacher vom Dienst?"

„Nein, das Ganze ist sehr fachlich. Es geht darum, nach Konsenz für große Ideen zu streben und die Interessen vieler Freiwilliger zu vereinen."

Der Vater hatte nun verstanden: „Das ist das Aal-Prinzip mein Junge, und das klingt sehr vernünftig."

„Was ist denn das Aal-Prinzip, Vati?"

„Andere arbeiten lassen."

Kapitel 7

Im Dialog: Recht, Politik und Open Source

„Der Einsiedler versündigt sich gegen die Gesellschaft, denn er entzieht ihr die Dienste, die er imstande wäre, ihr zu leisten."

– Rudolf von Jhering

Die GPL bringt es auf den Punkt

Martin Görnitz

(CC-Lizenz siehe Seite 563)

> „[T]he *GNU General Public License* is intended to guarantee your freedom to share and change free software, [. . .] the *GPL* clearly explains that there is no warranty for this free software, [. . .] the *GPL* assures that patents cannot be used to render the program non-free." (*GPLv3*, Zweiter Diskussionsentwurf)

Freiheit, Sharing, Haftung, Patente – man könnte meinen, die Autoren des folgenden Kapitels hätten sich an der Präambel zur neuen *GPL* orientiert, als sie ihre Beiträge für das Open Source Jahrbuch schrieben. Gleichzeitig werden die Relevanz und die Notwendigkeit der kritischen Behandlung dieser Themen bewusst, erwachsen doch daraus die Konflikte, deren Präsenz Entwickler wie Nutzer, Politiker, Bürger, Gerichte, Lobbyisten und Autoren gleichermaßen beschäftigen.

Die Tauglichkeit des Prinzips Freie-/Open-Source-Software (FOSS) in unserer Gesellschaft haben die vorangegangenen Kapitel nachdrücklich unter Beweis gestellt. Auf der anderen Seite zeigt die Notwendigkeit der wiederholten Präsentation der Erfolge, dass es mit der Bereitschaft der Gesellschaft, FOSS anzuerkennen und auf breiter Front zu integrieren, noch immer nicht befriedigend bestellt ist.

Die Basis einer solchen Integration findet sich auch im Gebrauch einheitlicher Begrifflichkeiten zur Definition der zugrunde liegenden Konzepte. Daher eröffnet Prof. Bernd Lutterbecks Beitrag „Sharing" den Einstieg zur näheren Bestimmung dieses oft verwendeten und selten näher bestimmten Begriffs, der ein Grundkonzept jedes offenen Modells bildet.

Eine logische Fortführung der Integration des FOSS-Modells in gesellschaftliche Strukturen offeriert Maika Büschenfeld. Ihr Artikel präsentiert eine zukünftige Koexistenz des demokratischsten aller Softwareentwicklungsmodelle mit den Ausprägungen elektronischer Demokratien und eröffnet so den Weg in eine offenere Gesellschaft.

Den freiheitlich geprägten Methoden und dem Potenzial zur Wandlung klassischer Strukturen zum Trotz droht die Gefahr der Inkompatibilität mit gesellschaftlichen Realitäten. Softwarepatente, das Widersprüchlichste, was die Gesellschaft dem FOSS–Modell entgegenstellt, sind weiterhin auf der Tagesordnung, nicht nur amerikanischer

Entwickler. Gerade die als *Trivialpatente* berüchtigt gewordenen Ausprägungen einer generalisierenden Vergabepraxis gefährden die Entwicklung internationaler FOSS–Projekte und erfordern proaktives Handeln. Einen diesbezüglich dem Modell zuträglichen Weg möchte die neue Version 3 der *GPL* einschlagen. Ob dies gelingen könnte, ergründet Lina Böcker in „Die GPL v3 – ein Schutzschild gegen das Damoklesschwert der Softwarepatente?".

Behindern die patentrechtlichen Widerstände bereits den Entwickler in seiner Arbeit, so verhindern häufig Unklarheiten über potenzielle Haftungsfragen den Eintritt in manchen Markt und damit auch die ungehinderte Ausbreitung des FOSS-Modells. Ein Vertreter von *SAP* benannte gegenüber dem Autor dieser Zeilen genau diese fehlende Klärung grundlegender Haftungsfragen als Hauptgrund für das zurückhaltende Engagement seines Unternehmens im FOSS Bereich. Christian Laux und Jan Widmer widmen ihren Beitrag „Produkthaftung für Open Source Software?" der Darstellung des Problems und geben praxisbezogene Einblicke in die Lösungsmöglichkeiten, welche sich dafür ergeben.

Als Beispiel für die Hindernisse, die sich dem FOSS-Modell auf der Spur zum gesamtgesellschaftlichen Erfolg in den Weg stellen, dient der Artikel von Leonhard Dobusch. „Von der Kommune zur Community: Freie Netze und freies Wissen auf der lokalpolitischen Agenda" beschreibt die Probleme und Handlungsmöglichkeiten auf kommunaler Ebene.

Ein zentrales Thema liegt allen Autoren dieses Kapitels gemeinsam am Herzen: FOSS kann der Gesellschaft viel mehr bieten als nur „kostenlose Software". Einem übergreifenden Erfolg des Modells stehen jedoch noch eine große Anzahl verschiedentlicher Probleme gegenüber, deren Lösung nicht zuletzt die Mitarbeit und die bereitwillige Unterstützung breiter Teile der Gesellschaft erfordern. Denn erst mit der Lösung dieser Probleme kann das Prinzip FOSS seine Vorteile voll ausspielen und an der Prägung und Formung der Gesellschaft zu ihrem Vorteil in vollem Maße mitwirken.

Sharing – Ein Kampf ums Recht

BERND LUTTERBECK

(CC-Lizenz siehe Seite 563)

Der Begriff *sharing* wird in englischen wie deutschen Texten schon fast inflationär benutzt, mal mit Übersetzung, mal ohne. Für den Normalgebrauch wird ein Begriffsinhalt vorgeschlagen, der aus der Anthropologie stammt und ihn gegen andere Verwendungsweisen abgrenzt. Diese Festlegung erweist sich für ökonomische und juristische Zusammenhänge als schwierig, aber letztlich fruchtbar. Der Beitrag entwickelt eine Typologie des *sharing* in verschiedenen Rechtsordnungen und stellt *sharing* als grundlegendes Konzept der Informatik vor. Darauf aufbauend wird ein Vorschlag vorgestellt, wie die deutsche Rechtsordnung mit dem Typ des *sharing* umgehen könnte. Es wird festgestellt, dass es ein Spannungsverhältnis zwischen dem auf Offenheit hin angelegten Typ des *sharing* und der Kontrollierbarkeit durch Dritte gibt. Es ist unklar, wie dieses Spannungsverhältnis aufzulösen ist.

Schlüsselwörter: information sharing · Public Domain · benefit sharing · sharecropping · car pooling · time sharing · file sharing

1 Einleitung

> „In essence, [...] open source involves two things: putting spare capacity (geeks surplus time and skill) into economic production; and sharing." (Economist 2005)

Wenn *sharing*, wie der *Economist* nahelegt, das Kernkonzept von Open Source darstellt, ist es gewiss nicht trivial, die Bedeutung dieses Begriffs in seiner ganzen Breite zu erfassen. Das ist in der Tat nicht so einfach, denn inzwischen ist sein Gebrauch fast inflationär. So verzichtet mittlerweile kaum ein Community-Portal darauf, sein Anliegen in den Dienst des *sharing* zu stellen. So z. B. das Portal für Bilder *flickr.com*: „The best way to store, search and share your photos". *Creative Commons (CC)* definiert sich als: „non-profit organization devoted to expanding the range of creative

work available for others legally to build upon and share." Offensichtlich ist *sharing* so grundlegend, dass es schon in der Präambel der *GNU Public License* von 1991 heißt:

> „The licenses for most software are designed to take away your freedom to share and change it."

In der (nicht offiziellen) deutschen Übersetzung wird daraus der Satz:

> „Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, die Software weiterzugeben und zu verändern."

Diese Übersetzung überrascht. Zumindest findet sich in keinem der gebräuchlichen Lexika diese Bedeutung *weitergeben* für *to share*. Wenn eine so wichtige Lizenz eine diskussionsbedürftige Übersetzung anbietet, dann spricht Manches dafür, den Typ *sharing* etwas gründlicher zu untersuchen. Der deutsche und der englische Text drücken offensichtlich nicht das Gleiche aus. Dies ist wohl unter anderem der Tatsache geschuldet, dass die deutsche Sprache nur wenige Begriffe hat, die gemeinschaftliche Aktivitäten ausdrücken. Solche Schwierigkeiten könnten auf sich beruhen, hätte nicht das jeweils nationale Recht einen legitimen Anspruch, die Bedeutung jeweiliger Begrifflichkeiten festzulegen – und damit natürlich die beteiligten Interessen in seinem Sinne festzuschreiben. Kurz gesagt: Begriffsfragen sind immer auch Machtfragen.

Die Situation wird auch dadurch nicht einfacher, dass inzwischen staatliche deutschsprachige Dokumente das englische Wort *sharing* benutzen, so etwa der Leitfaden *Information Sharing* der BundOnline-Initiative (Biggeleben und Schlagheck 2004). Oder das Konzept der *Shared Services* der neu gegründeten Bundesstelle für Informationstechnik (Klocke 2006). Es gibt also nicht nur – vielleicht problematische – Übersetzungen ins Deutsche, sondern auch das englische Wort *sharing* in offiziellen deutschen Dokumenten. Fast immer wird seine Bedeutung nicht erklärt, sondern vorausgesetzt. Solche Unklarheiten könnten dahin führen, dass Konzepte, die in einer bestimmten Kultur funktionieren, ohne Ansehen der kulturellen Unterschiede in die deutsche Rechtsordnung übernommen werden. Das könnte bei den neuerdings im Bereich der Terrorismusbekämpfung diskutierten Konzepten des *information sharing* der Fall sein.

Meine These ist nicht spektakulär: Ich möchte dafür werben, vorsichtiger mit dem Konzept *sharing* umzugehen. Diese Vorsicht könnte helfen, einige Probleme beim Umgang mit „geistigem Eigentum", aber auch beim Datenschutz besser zu verstehen. Dazu will ich den Begriff *sharing* untersuchen, um darauf aufbauend eine erste Typologie des *sharing* vorzustellen.

Abbildung 1: The Pig and the Box (MCM 2006, Deckblatt)

2 Du sollst teilen!

> „The Pig and the Box is a modern fable that teaches kids and adults alike that sometimes, yes, it's a good idea to trust and to share." (MCM 2006)

Seit Sommer 2006 kursiert ein Comic für Kinder, der ihnen den Sinn von Open Source oder juristisch z. B. den CC-Lizenzen verständlich machen soll – ein Gegenentwurf zu entsprechenden Comics der Copyright-Industrien.[1] Ein Schwein findet eine Zauber-Box, mit der sich alles vervielfältigen lässt, auch Futter zum Essen. Da es arm ist, wittert es eine Chance, endlich genug zu essen zu haben und beutet alles für sich aus. Auch die Nachbarn kriegen natürlich den plötzlichen Reichtum mit und möchten den Kasten für ihre Zwecke nutzen. Schweinchen will das nicht, reich will es werden. Also stellt es Regeln auf, unter denen die Nachbarn den Kasten benutzen dürfen. Dieses „Geschäftsmodell" führt zu diversen Katastrophen. Ein ziemlich egoistisches

1 Hier hieß der Held Captain Copyright, der nach dem Vorbild von Superman gegen Raubkopierer kämpfte. Der Comic war am 10. November 2006 nicht mehr auf der offiziellen Website verfügbar: http://www.captaincopyright.ca.

Schwein muss lernen, dass es manchmal besser ist, mit anderen zu teilen, zu „sharen". Am Ende steht es viel besser da als vorher, weil es „shared" und bringt es so zu einigem Wohlstand und besten Beziehungen zu den Nachbarn.

Die Geschichte ist auch für ernsthafte Erwachsene interessant, gibt sie doch Hinweise für den Normalgebrauch des Wortes *sharing*. Sehr viele Kinder in vielen Gesellschaften werden erzogen, Dinge mit anderen zu teilen – eine Moral, die man mit der Muttermilch aufsaugt. Für diese Moral steht der Begriff *sharing*. *Sharing* heißt eine Beziehung zu Menschen aufbauen, indem man etwas mit ihnen teilt. Dies setzt offensichtlich Vertrauen voraus, eine Eigenschaft, die ihrerseits durch eine gelingende Erziehung erzeugt werden kann. Mit anderen Worten: *Sharing* hat eine positive Konnotation, die jedem seit seiner Kindheit im Prinzip klar ist.

Juristen und Politiker wissen natürlich um diese Bedeutung und machen sie sich zu Nutze. Was mag wohl deutschen Benutzern des Wortes *sharing* durch den Kopf gegangen sein, etwa bei den Formulierungen des Leifadens *Information Sharing* der BundOnline-Initiative (Biggeleben und Schlagheck 2004). Jedenfalls sind sie, zum Teil aus guten Gründen, an die vorgeschlagene Entfaltung des Begriffs nicht gebunden. Beide müssen aber ins Kalkül nehmen, dass die „normalen" Bürger dem Begriff des *sharing* wahrscheinlich die Bedeutung zugrunde legen werden, die sie als Kind erworben haben. Am jeweiligen Ende dieser „Begriffsskala" stehen zwei Typen strategischer Optionen:

- Man muss juristisch in Abrede stellen, dass eine bestimmte Aktivität den Tatbestand des *sharing* erfüllt. Dies erfordert unter Umständen einen hohen logischen und argumentativen Aufwand, dem nicht unbedingt jeder Mensch gewachsen ist.
- Man macht sich juristisch die positive Konnotation zu Nutze, um damit möglicherweise ganz andere Ziele zu erreichen. Man vertraut also darauf, dass die Menschen nicht genau nachfragen.

Für beide Extreme gibt es aktuelle Beispiele. Die erste Strategie haben sich zwei prominente Persönlichkeiten der USA zu eigen gemacht: Richard Parsons (2002), CEO von *Time Warner* und David Kendall (2002), früherer Rechtsberater von Präsident Clinton und 2002 Rechtsvertreter der Hollywood-Industrien. Beide haben im Streit um *file sharing* und den Download von Musikdateien für die entsprechenden Industrien Partei ergriffen. Da sie um die eher positive Einstellung der amerikanischen Bevölkerung zur Thematik wussten, mussten sie die Aktivität des *downloading* aus dem positiven Bereich des *sharing* herausdefinieren:

> „A word here about terminology. The word 'file-sharing' is a euphemism and a serious misnomer. We are taught from our earliest years that sharing is good, and it generally is, but banks would have a problem if you walked in and wanted to share cash reserves in the teller's tills.

> Likewise, sharing of copyrighted materials is a *bad* kind of sharing if you have no right to distribute and copy the music or the movie. In fact, it's not really sharing at all, because if I share a piece of cake with you, we're each doing with a little less—I have half a piece and you have half a piece. This doesn't digital distribution since I don't lose anything by 'sharing' with you, and an infinite number of copies can be made because once something is disseminated on the Internet, you are going to exponentially increase the number of file copies. It's more accurate to call it file 'squaring' than file 'sharing'." (Kendall 2002, S. 8)

Juristen aller Kulturen wissen um die Macht der Begriffe und streiten deshalb um ihre Bedeutung. Bei David Kendall kommt der Antrieb für seinen terminologischen Aufwand sehr schön zum Vorschein: *Sharing* sei ein Wort, dem im Grunde jedes Kind die Konnotation „gut" gebe. Also, so muss man schlussfolgern, ist es die juristische Aufgabe, der Aktivität „Download" das Gute zu nehmen und zu „beweisen", dass sie in Wahrheit schlecht, also nicht *sharing* ist.

Die Juristin Jessica Litman (2004) hat aufgepasst und diese Strategie in einem großen Aufsatz offengelegt: *Sharing and Stealing*. Kendall benutzt Kekse als Methapher für rechtswidriges Tun. Er verschiebt die Bedeutung von *to share* in die Richtung auf *to part*, teilen. So wird ein ganzer Keks zu zwei halben, nur dann *teile* (*share*) man. *Sharing* hat dann immer etwas mit *verlieren* zu tun. Folglich müsse der Peer-to-Peer-Download rechtswidrig sein, man verliere ja schließlich nichts. Natürlich ist das Unsinn, ein klassischer logischer Fehler (*petitio principi*). Es ist ja gerade das Kennzeichen des sog. „geistigen Eigentums", dass es sich durch Weitergabe nicht verbraucht. „When I share my ideas, I don't loose anything", sagt Frau Litman. Den Unterschied machen sich alle Rechtsordungen zu Nutze, indem sie unterschiedliche Regeln für das Eigentum an Sachen und das „Eigentum" an Ideen aufstellen. Es wäre dann zu beweisen, warum der Austausch von Ideen den Regeln über das Sacheigentum unterliegen soll. Den Beweis bleibt Kendall schuldig.

Die zweite Strategie führt in die Mitte gegenwärtiger Innenpolitik in den westlichen Industrienationen: die Bekämpfung des Terrorismus. In Deutschland streiten die Parteien um den Aufbau einer gemeinsamen Antiterrordatei,[2] in den USA hatten die gleichartigen Diskussionen schon eine gesetzgeberische Konsequenz: den *Intelligence Reform And Terrorism Prevention Act of 2004* mit einer eigenständigen Sektion *Information Sharing*[3]. Auch in den USA geht es um die gemeinsame Nutzung informationstechnischer Ressourcen. Dies erfordert viel Geld, aber auch die

2 In der deutschen Gesetzessprache wird *information sharing* zu einem „Entwurf der Bundesregierung für ein Gesetz zur Errichtung gemeinsamer Dateien von Polizeibehörden und Nachrichtendiensten des Bundes und der Länder (Gemeinsame-Dateien-Gesetz vom 20. September 2006)" – eine in der deutschen Rechtssprache ungewöhnliche Benennung. Der Entwurf ist am 1. Dezember 2006 als Gesetz verabschiedet worden, vgl. Plenarprotokoll der 71. Sitzung v. 1. 12. 2006 und Bundestagsdrucksachen 16/2950, 16/3292 und 16/3642.

3 Siehe U. S. Congress (2004, S. 28).

Einschränkung von Freiheitsrechten. Hier benutzt der Gesetzgeber die positive Konnotation von *sharing*, um sich von der Bevölkerung die Legitimation zu beschaffen. Gleichzeitig setzt er die teilweise widerwilligen Bürokratien unter Druck, sich dem hehren Ziel nicht zu verweigern: „Wir müssen Informationen *teilen*, um den Kampf gegen den Terrorismus zu gewinnen." „An etwas Gutem kann doch nichts Schlechtes sein", wird man dem allgemeinen Volkswillen unterstellen dürfen.

Solche begrifflichen Entscheidungen erleichtern zumindest die Durchsetzung politischer Ziele. In der zynischen Variante, die hier keineswegs unterstellt wird, benutzen Politiker und Juristen das positiv besetzte Wort, um etwas nicht ganz so Gutes durchzusetzen. Ein Beispiel für eine durchaus positive juristische Strategie bei der Benutzung positiv besetzter Begriffe war die Entscheidung der Clinton-Administration, einen *Information Super Highway* aufzubauen. Was für einen deutschen Beobachter vielleicht ein wenig lächerlich klingen mag, hat für einen englischen Muttersprachler einen ganz anderen Klang: den Klang von Weite, Eroberung der offenen Prärie, von superschnellen Postkutschen und schließlich den Bau der staatsübergreifenden Autobahnen. Unzählige Western beschäftigen sich mit diesem amerikanischen Traum vom „weiten Land".[4] Diesen positiven Klang hatte sich schon der Vater Al Gores zu Nutze gemacht, der das Infrastrukturprojekt *Superhighway* für alle Amerikaner durchgesetzt hatte. Allerdings für Autos. Bei dem Wort *Information Superhighway* haben viele Amerikaner an diese positiven Effekte gedacht, die der Vater erreicht hatte. Sein Sohn hat sich dieses positive Gefühl klugerweise zu Nutze gemacht und für das Internet den Applaus eingeheimst. Möglicherweise war es auch diese Begriffspolitik, die die amerikanische Internet-Politik zum Erfolg gemacht hat.

Halten wir fest: „Du sollst teilen", bringen wir unseren Kindern bei. Offensichtlich sind wir dabei so erfolgreich, dass auch die Repräsentanten mächtigster Interessen vor den Konsequenzen dieser Erziehung Angst haben.

3 Typologie des sharing

Juristische Konzepte, die das Wort *sharing* verwenden, gibt es nicht erst seit dem Internet-Zeitalter. Im 19. Jahrhundert haben einige Südstaaten der USA eigene *Sharecropping Acts* erlassen. *Sharecroppers* bewirtschafteten ein Stück Land gegen einen Teil des Ertrages daraus. Sie besaßen auch meist keine Werkzeuge. Zugrinder etc. mussten sie also von den Landbesitzern mieten. Alles in allem ein Teufelskreis, da der eigene Ertrag meist nicht mal für Saat und Werkzeuge etc. des nächsten Jahres ausreichte, und sie so immer tiefer in Schulden gerieten. Die Folge dieser Gesetze war eine fast vollständige Verarmung der Landbevölkerung, wie Rivoli (2006, S. 49 ff.) berichtet. Zumindest die überwiegend schwarze Bevölkerung wird diesem Gebrauch von *sharing* wenig Gutes zuschreiben wollen. Strategisch verweist diese Verwendung von *sharing* wohl eher auf die zweite Strategie.

4 Vgl. die Ähnlichkeiten mit dem Begriff *Public Domain*, unten Fußnote 10.

Abbildung 2: Ben Shahn: Cotton Pickers (1935)[5]

Juristen brauchen Typen abstrakt formulierter Handlungen, weil sie nur dann rechtliche Folgen zuweisen können. Ein klassisches Beispiel ist das Recht für Verträge. Typische Handlungen wie Geben, Nehmen und Bezahlen werden ihrer Äußerlichkeiten entkleidet und kehren dann in Festlegungen etwa des Bürgerlichen Gesetzbuchs wieder. Diese Vertypungen benutzen die Juristen dann, um ihre logischen Deduktionen vorzunehmen. Es ist deshalb von ausschlaggebender Bedeutung, wie man einen Typ setzt, nur deshalb habe ich mich so umfänglich mit den Ausführungen von Kevin Kendall auseinandergesetzt.

Ich wollte mir einen Überblick verschaffen, wie verschiedene Rechtsordnungen den Begriff *sharing* benutzen. Ich bin dabei auf eine Vielfalt gestoßen, die ich nicht erwartet hatte.[6] Tabelle 1 fasst das Ergebnis meiner Recherche zusammen.

Eine Sortierung nach Wissensdomänen ergibt folgende grobe Schwerpunkte:

1. Besondere vertragliche Gestaltungen
   - *sharecropping/sharemilking*
   - *car sharing/car pooling*
   - *production sharing*
   - *Shareware*
   - *benefit sharing*

5 http://www.english.uiuc.edu/maps/poets/a_f/brown/photos.htm

6 Die Typologie ist sicher noch unvollständig, man denke etwa an ökonomische Begrifflichkeiten wie *market share* und *shareholder*.

2. Informatikkonzepte zur Verwaltung von Ressourcen
   - *time sharing*
   - *file sharing*
   - *spectrum sharing*
3. IT-Strategien
   - *information sharing*
   - *Shared Services*
4. Wissenschaftliche Konzepte
   - *shared mental models*
   - *sharing economy*

Es wäre eine wichtige Aufgabe, aus diesem empirischen Befund den Typ des *sharing* zu erstellen. Das kann ich hier nicht leisten. Ich will lediglich auf drei, für die heutige Diskussion wesentliche Typen näher eingehen: *information sharing*, *car sharing/pooling* und *benefit sharing*.

Seit einigen Jahren ist der Begriff *information sharing* in Gebrauch. Die amerikanische Rechtssprache verwendet ihn, allerdings findet sich keine Definition, sondern lediglich eine prozedurale Umschreibung. Erst wenn man die offizielle Literatur zu 9/11 in Hearings und Berichten zu Hilfe nimmt, ergibt sich ungefähr folgender Begriffsinhalt: Der offizielle *9/11 Commission Report* (2004) stellt als größtes Hindernis für Fortschritte auf dem Gebiet der Terroristenbekämpfung „the human or systemic resistance to sharing information" heraus (S. 416). Die Anreizstruktur für die Nutzung von Informationen sei völlig falsch gesetzt. Die Weitergabe werde bestraft, das *sharing* aber kaum belohnt. Der entscheidende Satz lautet dann:

> „There are no punishments for *not* sharing information. Agencies uphold a 'need-to-know' culture of information protection rather than promoting a 'need-to-share' culture of integration." (S. 417)

Information sharing heißt deshalb hier: integrierte Datenverarbeitung nach zu definierenden technologischen Standards plus Etwas, das die Nutzer der Informationen gemeinsam haben (sollten): etwa die gemeinsame Bedrohung, die Verteidigung spezieller amerikanischer Werte und dergleichen. *Information sharing* ist also in der amerikanischen Rechtssprache kein rein datentechnischer Begriff. Geradezu arglos und technokratisch erscheint der Zugriff der schon mehrfach zitierten BundOnline–Initiative:

> „Information Sharing wird im Rahmen dieses Leitfadens definiert als der Austausch von Informationen zwischen IT-Systemen, in der Regel

Sharing-Typ	Wissensdomäne	Rechtliche Ausprägung	Literatur
time sharing	Informatik (Betriebssysteme); Mietrecht (z. B. Ferienwohnungen)	Mietrecht (§§ 481 ff. BGB)	Strachey (1959); McCarthy (1983); Licklider (1960); Lee (1992*b*); Meyers Lexikon online (2006)
file sharing	Informatik; P2P-Netzwerke	rechtl. Konzeptualisierung; Fallrecht zum Urheberrecht	Böhm (2005); Litman (2004); Brinkel (2006)
information sharing	administrative Datenverarbeitung; Terrorismusbekämpfung; Datenschutz	U.S. Congress (2004); Gemeinsame-Dateien-Gesetz (Entwurf 9/2006); Leifaden *Information Sharing*	Swire (2006); de Valk und Martin (2006); Biggeleben und Schlagheck (2004)
Shared Services	Vorfeldbekämpfung von Terrorismus und Kriminalität; administrative IT-Strategien	eGovernment-Initiative der Regierung des UK; Bundesstelle für Informationstechnik	www.cio.gov.uk/shared_services; Klocke (2006); www.bit.bund.de
spectrum sharing	Netzwerktheorie; Telekommunikation	Regulierung in Telekomgesetzen (Netzneutralität)	van Schewick (2004)
production sharing (Outsourcing)	internationale Ökonomie (Verlagerung von Phasen der Produktion; z. B. Ausbeutung von Ölressourcen)	Verträge zwischen Unternehmen, Herstellern und Eigentümern von Ressourcen	Kurz (2006)
sharecropping sharemilking	Produktion von Baumwolle (USA); Milchwirtschaft (Neuseeland)	*Sharecropping Acts* in Gliedstaaten der USA (19. Jh.); Verträge zwischen Sharemilkern und Besitzern von Herden (Gesetz in NZ v. 1937)	Rivoli (2006); www.fedfarm.org.nz
car sharing (*car pooling*)	gemeinsame Nutzung von Autos (Nutzung von Autos für gemeinsamen Zweck, z. B. in San Francisco: *Sharing* von Kosten)	Verträge zwischen Car-Sharing-Anbietern und Kunden (keine Rechtsvorschriften, sondern soziale Normen)	de.wikipedia.org/wiki/Carsharing; www.ridenow.org/carpool
Shareware	Vertrieb von Software	lizenzrechtliche Lösung *sui generis*	Gehring (1996)
space sharing	öffentlicher Nahverkehr; Raumordnung	Verkehrsrecht; Raumordnungsrecht	www.sharedspace.org
benefit sharing	Genetik; Nord-Süd-Kooperationen; Bewirtschaftung von Nationalparks (USA)	*Bonn Guidlines* „Access to Genetic Ressources and Benefit Sharing" von 2002; National Parks Omnibus Management Act v. 1998	www.biodiv.org; nature.nps.gov/benefitssharing
shared mental models	ökonomische (Institutionen-)Theorie	Rechtstheorie; Institutionenökonomik	Denzau und North (1994)
sharing economy	Netzwerkforschung; Juristische Institutionenlehre	rechtl. Konzeptualisierung	Benkler (2004); Benkler (2006)

Tabelle 1: Sharing-Typen

> mit dem Ziel der Darstellung dieser Information auf einer Website." (Biggeleben und Schlagheck 2004, S. 7)

Der deutsche Streit um die sog. „Antiterrordatei" zeigt, wie stark diese gemeinsame Nutzung von Informationen in unterschiedliche Ideologien verstrickt ist und damit den Datenschutz an seine Grenzen bringt. Denn historisch war der Datenschutz gewissermaßen ein Gegenentwurf zur integrierten Datenverarbeitung, entsprechend ist „Zweckbindung" eines der wichtigsten Prinzipien des Datenschutzes. Leitbild war die getrennte und nicht die „geteilte" Nutzung jeweiliger Datenbestände.

Ein anderes Verständnis von *information sharing* setzt auf der empirisch beobachtbaren Nutzung des Internets durch viele Menschen auf der ganzen Welt auf: Nur wenige benutzen noch klassische Lexika, Telefonbücher oder dergleichen. Man benutzt *Google*, *Wikipedia* etc. – Wissen, das andere zur Verfügung gestellt haben. Und man stellt seinerseits Wissen ins Netz, das andere benutzen können – irgendwo auf der Welt. Jessica Litman (2004, S. 11) umschreibt deshalb ihr Verständnis von *information sharing* so:

> „[W]e're not each downloading facts from some giant Encyclopedia Britannica in the sky. We are both finding what we need and making available material that we've generated or assembled."

Am Begriff *information sharing* zeigt sich also ein zweifaches Problem: Ziemlich weit auseinander liegende Bereiche werden zum einen mit dem gleichen Begriff bezeichnet. So etwas ist juristisch immer bedenklich, weil man nicht weiß, was man genau erwarten muss. Zum anderen ist es für die von Prof. Litman bezeichneten Aktivitäten völlig unklar, ob man sie überhaupt mit den Mitteln des Rechts erfassen muss.

Für die juristische Typenbildung besonders interessant ist die Unterscheidung von *car sharing* und *car pooling*, die im Amerikanischen gemacht wird. *Car sharing* ist eine Austauschform, die auch in Deutschland ohne besondere Mühe den Regeln des Bürgerlichen Gesetzbuchs zu unterwerfen ist. Fast jede deutsche Großstadt hat ja Unternehmen, die nach diesem Modell arbeiten. Bei *car pooling* versagt diese typologische Einordnung. Hier werden nicht Autos, sondern Passagiere zusammengefasst. Ab einer bestimmten Anzahl von Passagieren dürfen die Autos besonders gekennzeichnete Fahrbahnen bevorzugt benutzen. Wie u. a. Benkler (2004, S. 281 ff.) berichtet, funktionieren *car pools* ohne rechtliche Regelungen, sondern alleine aufgrund sozialer Normen, die eingehalten werden. Manchmal, aber nicht notwendigerweise, findet ein *sharing* statt, indem sich die Insassen eines Autos die Kosten teilen. In diesem Typ beruhen die Handlungen der Akteure nicht auf dem Prinzip von Leistung und Gegenleistung, können also nur außerhalb des Vertragsrechts rechtlich erfasst werden. Dies ist also ein Beispiel für einen Typ, in dem das Wort *sharing* gar nicht verwendet wird, wohingegen das rechtliche Institut, das das Wort verwendet, nicht im eigentlichen Sinne *sharing* ist.

Abbildung 3: Car-pooling-Verkehrsschild in Norwegen[7]

Soziale Normen bringen *sharing* zum Funktionieren, so könnte man mit aller Vorsicht die Gemeinsamkeiten herausstellen. Die Gesetze zum *information sharing* können diese Normen natürlich nicht anordnen und sind selbst im Falle der Geheimdienste darauf angewiesen, dass Menschen und Institutionen mitspielen. Diese Tatsache macht es dann schwierig bis unmöglich, *sharing* mit den klassischen juristischen Typen zu erfassen. Ein schönes Beispiel ist wiederum der schon zitierte Kevin Kendall. Er ist gewiss als früherer Rechtsberater eines amerikanischen Präsidenten ein herausgehobener Jurist. Aber er muss eine geradezu verzweifelte Logik bemühen, um die Gleichung *sharing* = *stealing* aufzustellen. Kendall muss dies tun, weil seine Sicht immer ein Verhältnis von Leistung und Gegenleistung herstellen muss, um die jeweiligen Sphären der Verantwortung abgrenzen zu können. Einen anderen Typ hat diese herrschende Sicht ja nicht. Sie kennt nur die Regelung durch den Staat und durch Vertrag.

Guido Brinkel, der in Deutschland das maßgebliche juristische Werk über *file sharing* geschrieben hat, steht Kendall in seiner Logik in nichts nach:

> „Durch diese [Peer-to-Peer-Technologie] wird der weltweite Austausch von Daten zwischen sich ansonsten völlig unbekannten Personen ermöglicht. Der Begriff Tausch ist dabei missverständlich, weil die Übertragung der Daten nicht zwingend auf *Gegenseitigkeit* beruht und die Werke bei der Übertragung nicht für den jeweiligen Anbieter *verlorengehen*, sondern der Empfänger von angebotenen Dateien vielmehr Kopien anfertigt. File sharing ermöglicht insoweit eher das *Teilen* bestimmter Daten [. . .]." (Brinkel 2006, S. 17)

Auch Yochai Benkler hat diese juristische Rabulistik zurückgewiesen und stattdessen eine ökonomisch fundierte „Theorie des Sharing" vorgeschlagen (Benkler 2004) –

7 Das Schild heißt wörtlich *Sambruksfelt* (Zusammennutzungsfeld) und ist seit Sommer 2006 im Gebrauch, siehe http://www.vegvesen.no/vegskilt/opplysningsskilt/index.stm.

allerdings ist die empirische Basis noch schwach (vor allem *car pooling* und *distributed computing*). Für ein offensichtlich ungelöstes Problem der Typenbildung übernimmt Benkler die Ergebnisse der modernen Ökonomie, die sich als Sozialwissenschaft versteht[8], und wendet diese Ergebnisse auf die juristische Typenbildung an. Seine Arbeiten haben ihm in den USA den Ruf des bedeutendsten (juristischen) Theoretikers der Informationsgesellschaft eingebracht. In seinem Verständnis ist *sharing* eine nachhaltige soziale Praxis („sustainable social practice"). „Social sharing", so seine grundlegende These, „represents a third mode of economic production, alongside markets and the state." Die dieser Sicht zugrunde liegende ökonomische Rationalität wird in der Definition bei *Wikipedia* deutlich:

> „Sharing is the joint use of a resource. In its narrow sense, it refers to joint or alternating use of an inherently finite good, such as a common pasture or a timeshared residence. In a broader sense, it can also include the free granting of use rights to a good that is capable of being treated as a nonrival good, such as information. Still more loosely, 'sharing' can actually mean giving something as an outright gift: for example, to share ones food really means to give some of it as a gift."[9]

Eingangs hatte ich den *Economist* mit seinen zwei Essentials von Open Source zitiert: Knappe Ressourcen an Fähigkeiten plus *sharing*. *Sharing* ist eine ökonomische Grundmodalität und funktioniert aufgrund von sozialen Normen, die man schon als Kind erwirbt. Das Beispiel *car pooling* zeigt aber, dass *sharing* kein Typus sein muss, der im Gegensatz zu gesetzlichen Maßnahmen steht. Hier schafft ja der Staat eine Infrastruktur und hofft darauf, dass seine Maßnahmen Aktivitäten des *sharing* anregen. In diesem Fall macht sich Verkehrspolitik *sharing* zu Nutze, bei *information sharing* ist es die Politik der Inneren Sicherheit mit ihren Gesetzen.

Benefit sharing, der dritte Typ, den ich hier vorstelle, schützt das Gefüge sozialer Normen des *sharing* vor der Ausbeutung durch kommerzielle Interessen Dritter: Viele sich entwickelnde Staaten sind reich an natürlichen Ressourcen, die das Interesse von internationalen, zumeist amerikanischen Unternehmen wecken. Solche Pflanzen, Tiere und Mikroorganismen sind das gemeinsame Erbe der Menschheit, rechtlich gesprochen befinden sie sich in der Public Domain[10]. Jeder Mensch darf sie also nutzen. Die Unternehmen können aus diesen Ressourcen z. B. Arzneimittel entwickeln, die

8 Vgl. meinen Beitrag „Die Zukunft der Wissensgesellschaft" (Lutterbeck 2006), in dem wichtige Ergebnisse dieser modernen Forschungen referiert werden.

9 http://en.wikipedia.org/wiki/Sharing

10 In der amerikanischen Rechtssprache bezeichnet *Public Domain* die Gesamtheit des Wissens, das nicht dem Urheber- oder Patentrecht unterliegt – eine Art Gegenstück dieser Rechte. Wenn etwa die Geltungsdauer des Urheberrechts ausläuft, fallen die geschützten Werke in die Public Domain. Auch dieser Begriff ist nur schwer in die deutsche Rechtssprache übersetzbar. Meist wird der Begriff synonym mit dem Begriff *commons* oder Allmende benutzt. *Wikipedia*, http://en.wikipedia.org/wiki/Public_domain, weist darauf hin, dass manchmal – irrtümlich und umgangssprachlich – auch solche Werke zur Public Domain gezählt werden, die dem Urheberrecht unterliegen. Dazu zählen alle GPL-Lizenzen und

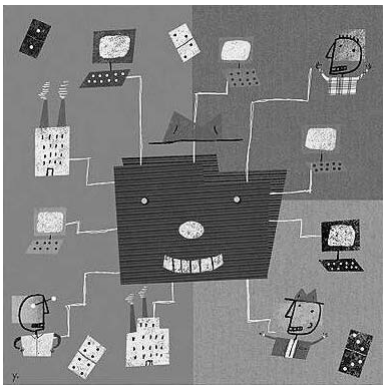


Abbildung 4: James Yang: File Sharing (digital) / Images.com

dann patentiert werden und in den Herkunftsländern Lizenzgebühren verursachen, die die zumeist armen Länder nicht aufbringen können.[11] Der Typ des *sharing* wird hier gewissermaßen aufgesplittet: Im nationalen Kontext des einzelnen Landes bleibt es beim Modus des *sharing*. Die jeweils geltenden sozialen Normen sorgen dafür, dass die Menschen eines Landes etwa die Heilkraft einer bestimmten Pflanze *so* nutzen können, dass sie sich alle teilen – ohne irgendwelche Lizenzgebühren. Im Verhältnis zu Dritten, etwa amerikanischen Konzernen, erhält die Public Domain eine Art Eigentumsstatus, die den „Eigentümer" berechtigt, von den Dritten Kompensation für das *sharing* von Ressourcen zu verlangen.

Natürlich spricht nichts für die Annahme, dass diese Normen der Menschen erst mit dem Internet entstanden sind. Klärungsbedürftig ist jedoch, warum sie gerade jetzt so wichtig geworden sind. Ich behaupte, dass die moderne Informatik auf diese Frage eine Antwort geben kann.

Wikipedia selbst. Bei GPL-lizenzierten Werken wird ja nur auf die Durchsetzung der Urheberrechte verzichtet. Wenn man dieses enge Verständnis von Public Domain beibehält, wird es schwierig, Worte für den Raum zu finden, der durch die *GPL* und viele neu entstandene Wissensbestände entstanden ist. Um diese Schwierigkeiten zu vermeiden, schlage ich eine Definition von Public Domain vor, die auf den Zugriff zur Ressource abstellt. Deshalb verwende ich eine Definition der amerikanischen Juristen Anupam Chander und Madhavi Sunder (2004, S. 1338): „Ressources for which legal rights to access and use for free (or to nominal sums) are held broadly. [. . .] [It] refers to ressources to which there are rights of access shared among *all* people". Eine solche Definition macht es dann leicht möglich, z. B. auch die von den Autoren behandelten Probleme der unfairen Ausbeutung von Ressourcen juristisch zu fassen und GPL-lizenzierte Werke der Public Domain zuzuschlagen.

11 *Benefit sharing* ist als Instrument z. B. in Regulierungen über die Nationalparks der USA festgehalten (National Parks Omnibus Management Act of 1998, 16 USC Sec. 5935). Interessanterweise haben aber die USA die internationalen Verträge, die den armen Ländern ein *benefit sharing* ermöglichen sollen, nicht ratifiziert, vgl. die Website der „Convention on Biological Diversity", http://www.biodiv.org/world/parties.asp.

Abbildung 5: Giampiero: Vater & Sohn / aboutpixel.de

4 Sharing – die gemeinsame Nutzung knapper Ressourcen

> „The sharing of information is the most important type of resource sharing." Licklider und Vezza (1978)

Konzepte des *sharing* gehören seit langem zur fachlichen Domäne der Informatik: *time sharing* seit den 60er, *file sharing* seit den 90er Jahren. Informatiker müssten also ein klares, auch wissenschaftlich begründetes Verständnis von *sharing* haben. Um dies herauszufinden, habe ich vor einigen Monaten einen kleinen Kreis (16 Personen) teilweise weltberühmter Informatiker nach ihrem Verständnis von *sharing* gefragt. Hierzu habe ich ihnen zwei Bilder aus den Datenbanken *Images.com* und *aboutpixel.de* gezeigt.

Das erste Bild (aus der Datenbank Images.com) zeigt *file sharing* in einer Graphik (siehe Abbildung 4). Mit dieser Darstellung war der Kreis einverstanden. Die Graphik drücke ganz gut den Gedanken aus, dass die eine Ressource Computer zu gleicher Zeit von vielen Entitäten genutzt werden kann. Danach habe ich ein Photo (aus der Datenbank aboutpixel.de) gezeigt. Ein Vater und sein Kind lächeln sich an (siehe Abbildung 5). Dieses Bild hatte ich durch ein *related keyword* „Happy smiling" gefunden, das die Datenbank zum Thema *file sharing* anbot.

Diese Such-Heuristik war auch für mich überraschend. Ich habe die Logik der Recherche an die Teilnehmer weitergegeben: Was haben die Abbildungen miteinander gemeinsam? Nichts, so die Antwort von Fünfzehn. Allein die mathematische Berühmtheit dieses Kreises war sich als Einziger sicher: „Diese Bilder meinen das Gleiche", jedes drücke auf seine Weise den Gedanken des „sharing" aus: „Entitäten teilen sich eine Ressource zur gemeinsamen Nutzung untereinander auf. Ein Vater und sein Kind teilen eine gemeinsame Freude."

Heutige Computernutzer als auch Programmierer können sich die primitiven Umstände der Frühzeit sicher kaum mehr vorstellen: Rechenzeit war extrem knapp und teuer, es gab nur wenige Computer für viel zu viele Interessenten. Der Alltag war eher lästig, man musste seine Lochkartenstapel bei der Zentrale abgeben, um sie, häufig genug erst einen Tag später, mit oder ohne Fehlermeldungen zurückzuerhalten. Es gab zur gemeinsamen Nutzung der knappen Ressource Computer aus ökonomischen Gründen keine Alternative. Die frühe Informatik hatte sich deshalb vor allem drei Ziele gesetzt:

1. Entwicklung von Systemen, die die knappe Ressource Computer optimal verwalten. Dies führte zur Entwicklung der ersten „Time-Sharing-Systeme", einer Methode, mit der möglichst viele Programme zur gleichen Zeit ablaufen konnten (um 1960).

2. Entwicklung von Techniken, mit der die Benutzer den Computer interaktiv, ohne das Hinzutreten weiterer Instanzen benutzen konnten („interactive computing", nach 1960).

3. Verbindung entfernter Orte durch technische Netze (ab Mitte 1960).

Die Entdecker der neuen Betriebssystem-Konzepte waren herausragende Ingenieure, aber sicher keine Personen, die an Begriffsfragen irgendeine Aufmerksamkeit verschwendet haben. Überdies ist die Urheberschaft für das neue Konzept und den neuen Begriff *time sharing* noch immer nicht ganz geklärt (Lee 1992*b*,*a*). Sicher scheint nur, dass die Nutzer dieser Systeme schon bald miteinander kommunizierten und eine neue Erfahrung machten: Man konnte sehr schnell mit Gleichgesinnten seine Programme verbessern und Erfahrungen austauschen. Die technischen Talente waren nicht notwendig vor Ort, es lag daher nahe, sie durch technische Netze miteinander zu verbinden. Auch die Vernetzung war also zunächst ein ökonomisches „Muss" (Lehtisalo 2005, S. 18).

Das für viele Überraschende war indessen, dass diese Betriebssystem-Innovation die Rechenmaschine Computer durch die Benutzung von einem „blöden Rechenknecht" in ein neues soziales Medium verwandelt hat. Denn inzwischen wollten nicht nur Programmierer, sondern auch „ganz normale" Laien Computer für sich nutzen. Man muss sich das wohl als eine Art evolutionären Prozess vorstellen, in dem die erlernte Fähigkeit zum *sharing* mit den neuen technischen Möglichkeiten verbunden wird. Und die Verbindung schafft dann wieder neue Möglichkeiten, an die vorher niemand gedacht hat.

Diesen Umschlag einer technisch-ökonomischen Entwicklung in eine soziale Innovation hat wohl niemand präziser vorhergesehen, aber auch selber gestaltet als Joseph C. R. Licklider. Er war ein herausragender IT-Stratege innerhalb des *Department of Defence* und einer der Vordenker des wichtigsten Projekts, dem Arpanet. Als gelerntem Psychologen war ihm mehr als den meisten anderen Pionieren die völlig neuartige

Qualität menschlicher Kooperation aufgefallen, die auf einmal praktiziert wurde. *Time sharing* hat er bereits 1960 als *thinking center* bezeichnet – als zentrales Konzept zur Verbindung von Mensch und Computer (Licklider 1960). In einem späten umfangreichen Aufsatz von 1978 hat er die Melange aus Innovationen bei Betriebssystemen, menschlichen Nutzungshandlungen und Netzwerk-Applikationen so gekennzeichnet:

> „Some of the problems and issues in network applications are mainly technical and some are mainly nontechnical, but almost all are mixtures of the two, and in most of them the technical and nontechnical factors interact strongly." (Licklider und Vezza 1978, S. 1330)

Auch wenn es nicht allen Informatikern bewusst ist: *Sharing* ist ein zentrales Konzept der Informatik. In einem ersten Schritt ist es die Umsetzung eines ökonomischen Prinzips in Technologie, eine Einsicht, die die zitierte Definition bei *Wikipedia* sehr gut zum Ausdruck bringt. In einem zweiten Schritt umfasst es die Veränderung des Artefakts durch soziale Nutzungshandlungen. Die Aufgabe der Informatik ist es jetzt, diesen Prozess der Evolution technischer Artefakte zu unterstützen und voranzutreiben.[12] Die Informatik gibt also dem Prozess des *sharing* eine Richtung.

5 Sharing – über Begriffspolitik hinaus

> „Peer to peer file-sharing platforms constitute an arena—a sphere that coordinates and regulates communicative activities of many individuals." (Pessach 2007, S. 26)

Sharing könnte eine anthropologische Konstante sein. Menschen haben sie entwickelt, weil sie in diesem Modus effizient und vertrauensvoll Güter und Ideen austauschen können. Die moderne Informatik setzt auf diesem Verhaltensmuster auf und macht so Benklers dritten Modus der Produktion möglich. Dieser Modus ist aber eingebettet in die vorfindlichen juristischen Institutionen – in erster Linie die Eigentumsordnung und das System der Grundrechte und Grundfreiheiten.

Dem Recht fällt die Aufgabe zu, diesem Modus Raum zu geben. Das war ja gerade der Geniestreich der *GPL*: Informatiker, nicht etwa Juristen, haben eine neue Institution erfunden, die auf der Basis des geltenden Rechts die Infrastruktur einer künftigen Zeit reguliert – eine soziale Innovation, die den Nobelpreis verdient hätte. So hat die *GPL* einen gesellschaftlichen Raum geschaffen, der sich als Public Domain[13] konstituiert. Die Kämpfe um Softwarepatente und P2P-Plattformen, um das Monopol von *Microsoft* und die Aktivitäten der Hollywood-Industrien geben vielleicht nur einen Vorgeschmack auf Kämpfe, die noch bevorstehen.

12 Sehr gute Beispiele für diese modernen Forschungsrichtungen sind die Arbeiten des Karlsruher Informatikers Klemens Böhm. Schon seine Überschriften zeigen, dass es um Konzepte geht, in denen Recht, Ökonomie und Informatik aufeinander bezogen werden müssen (Böhm 2005).

13 Zum Begriff Public Domain vergleiche man nochmals oben Fußnote 10.

Die ersten Runden haben zweifellos die Vertreter des *sharing* gewonnen – wenn auch nur nach Punkten. Es ist gewissermaßen der letzte Schrei, dem zumindest die führenden Vertreter der (zumeist amerikanischen) Wissenschaft gefolgt sind. In ihrer Euphorie haben sie die dunklen Seiten des Geschehens vielleicht etwas zu wenig beleuchtet: staatliche und private Kontrolle und die Ausbeutung von Minderheiten. Diese Blindheit könnte ihre Ursache in der binären Rhetorik haben, die sich viele Vertreter des Open-Source-Gedankens zu eigen gemacht haben: Hier geistiges Eigentum, da Public Domain; hier kommerziell, da *sharing*; hier die Bösen, da die Guten.[14]

In den nächsten Runden muss es darum gehen, die Bildung neuer Institutionen mit viel Phantasie voranzutreiben. Das können z. B. wie im Fall des *benefit sharing* völlig neuartige Hybride sein, die Eigentum und Allmende miteinander verbinden. Immer muss es darum gehen, die Grundelemente des *sharing*, als Grundelemente des Open–Source-Paradigmas, zu erhalten und, wo nötig, zu vergrößern. Die Grundelemente[15] sind:

- das Wissen in der Public Domain,
- Peer Review als Reputationsmechanismus und
- die Möglichkeit der Teilnahme durch jeden Menschen.

Die vielleicht größte Herausforderung in unserem Kulturkreis besteht darin, die Kultur der Offenheit mit den (legitimen) Kontrollbedürfnissen diverser privater und öffentlicher Institutionen in Einklang zu bringen. Der Verweis auf den Datenschutz könnte insofern trügen, als dass das gegenwärtige Konzept gerade nicht auf Offenheit hin optimiert ist und die gemeinsame Nutzung von Daten eher skeptisch sieht. Das Konzept sieht den Betroffenen eher als Monade, als eine untrennbare Einheit, die ihr Recht einsam gegen die Mächtigen erkämpft.[16] Es muss umformuliert werden in ein (gemeinsames) Recht auf Entfaltung in sozialen Räumen. Rechtlicher Anknüpfungspunkt ist insoweit nicht so sehr das Grundrecht auf freie Entfaltung der Persönlichkeit (Art. 2 Grundgesetz, GG, „Informationelle Selbstbestimmung"), sondern das Grundrecht der Informationsfreiheit (Art. 5 GG). *Free speech* ist das alles entscheidende Element des neuen Produktionstypus *sharing*.

14 Dies ist der Kern der Kritik von Chander und Sunder (2004). Auch in der neuesten Wissenschaft melden sich kritische Stimmen zum vielleicht zu optimistischen Ansatz von Benkler, siehe dazu Strahilevitz (2007) und Tushnet (2006).

15 Ich übernehme diese Elemente hier aus einem Aufsatz von de Valk und Martin (2006) über „Publicy shared intelligence", die darauf hinweisen, dass Offenheit als solche nicht hinreichend sei. Viel wichtiger sei die Notwendigkeit einer „triangulation of forums". Man nimmt mehrere Foren als Vermessungspunkte, um den Lösungsraum besser erfassen (triangulieren) zu können. Triangulation ist ein in den Sozialwissenschaften und von einigen Politikern benutzter Begriff. Er deutet z. B. an, dass mehr als eine Methode benutzt wurde, um ein Ergebnis zu erzielen. Immer geht es darum, eine einzelne, möglicherweise einseitige Sichtweise zu Gunsten einer breiteren Perspektive zu überwinden.

16 Das wissenschaftliche Konzept für den Datenschutz in Deutschland stammt von Wilhelm Steinmüller, Bernd Lutterbeck und Christoph Mallmann. Wer sich Zeit nimmt, kann es in Bundestagsdrucksache VI/3826 v. 7. September 1972 nachlesen.

Der amerikanische Jurist James Boyle hat für die amerikanische Diskussion die prägenden Metaphern für diese Art von Streitigkeiten formuliert (Boyle 2006). Er spricht von geistiger Landnahme und dem Ökosystem des digitalen Zeitalters. Wie schon erwähnt, ist die Vorstellung vom „weitem Land" prägender Bestandteil der amerikanischen Kultur. Auf die deutsche Diskussion passt eine andere Metapher vielleicht besser. Formuliert hat sie der deutsche Jurist Rudolf von Ihering: „Der Kampf ums Recht."[17] Dieses, in vielem noch zu findende Recht, wird darüber entscheiden, mit welcher Infrastruktur künftige Generationen ihren Reichtum und ihr Glück erzeugen werden.

Jetzt versteht man vielleicht: Es geht in diesem Beitrag nicht um eine blutleere akademische Betrachtung eines populären Modebegriffs. Es geht mir um die Standard- oder Default-Einstellung unserer Rechtsordnung. Die Open-Source-Bewegung kämpft für einen neuen Standard. In diesem Sinne ist *sharing* vor allem auch ein Kampf ums Recht.

Literatur

9/11 Commission (2004), The 9/11 Commission Report, Gutachten, National Commission on Terrorist Attacks Upon the United States, Washington. http://www.9-11commission.gov/report/911Report.pdf [01. Jan 2007].

Benkler, Y. (2004), 'Sharing Nicely: On Shareable Goods and the Emergence of Sharing as a Modality of Economic Production', *The Yale Law Journal* **114**(2), S. 273–358. http://www.yalelawjournal.org/pdf/114-2/Benkler_114-2.pdf [01. Jan 2007].

Benkler, Y. (2006), *The Wealth of Networks: How Social Production Transforms Markets and Freedom*, Yale University Press, New Haven und London. http://www.benkler.org/wealth_of_networks/index.php/Main_Page [19. Jan 2007].

Biggeleben, F. und Schlagheck, H. (2004), Information Sharing, Leitfaden, Kompetenzzentrum CMS, Bundesverwaltungsamt, Köln. http://www.bva.bund.de/cln_047/nn_951976/SubSites/BIT/DE/Shared/Publikationen/GSB__Publikationen/Information__Sharing,templateId=raw,property=publicationFile.pdf/Information_Sharing.pdf [01. Jan 2007].

Boyle, J. (2006), Eine Politik des geistigen Eigentums: Umweltschutz für das Internet?, *in* J. Hofmann (Hrsg.), 'Wissen und Eigentum – Geschichte, Recht und Ökonomie stoffloser Güter', Bundeszentrale für politische Bildung, Bonn.

Brinkel, G. (2006), *Filesharing. Verantwortlichkeit in Peer-to-Peer-Tauschplattformen*, Mohr Siebeck, Tübingen.

17 Von Ihering, der seinen Zeitgenossen als eine Art Weltwunder galt, hat diesen Vortrag 1872 in Wien gehalten. Die Tatsache, dass dieser Vortrag alleine in 26 Sprachen übersetzt wurde, gibt Hinweise auf die damalige Bedeutung seiner Ausführungen, vergleiche den einschlägigen Artikel bei *Wikipedia*, http://de.wikipedia.org/wiki/Rudolf_von_Jhering.

Böhm, K. (2005), 'Wer sorgt in Zukunft für Recht und Ordnung? Überlegungen zur Rolle von Peer-to-Peer Systemen in Technik und Gesellschaft', Präsentation. http://www.ipd.uka.de/Lockemann/AV.pdf [01. Jan 2007].

Chander, A. und Sunder, M. (2004), 'The Romance of the Public Domain', *California Law Review* **92**(5). http://papers.ssrn.com/sol3/papers.cfm?abstract_id=562301 [19. Jan 2007].

Denzau, A. T. und North, D. C. (1994), 'Shared Mental Models: Ideologies and Institutions', *Kyklos* **47**(1), S. 3–31. http://129.3.20.41/eps/eh/papers/9309/9309003.pdf [01. Jan 2007].

Economist, T. (2005), 'The economics of sharing', *The Economist* **3. Feb 2005**. http://www.economist.com/finance/displaystory.cfm?story_id=E1_PGNPSGN [01. Jan 2007].

Gehring, R. (1996), Freeware, Shareware und Public Domain, Studienarbeit, Informatik und Gesellschaft, Fachbereich Informatik, Technische Universität Berlin. http://ig.cs.tu-berlin.de/ma/rg/ap/1996-06/GehringRA-FreewareSharewarePublicDomain-13061996.pdf [01. Jan 2007].

Kendall, D. E. (2002), 'Copyright in Cyberspace', Brigance Lecture. http://www.copyrightassembly.org/briefing/DEKWabashSpeech4.htm [01. Jan 2007].

Klocke, J. (2006), 'Exklusiv-Interview mit Johannes Keusekotten, Leiter der Bundesstelle für Informationstechnik (BIT)', Interview. http://www.awv-net.de/cms/upload/awv-info/pdf/1-06-Interv.pdf [01. Jan 2007].

Kurz, C. J. (2006), Outstanding Outsourcers: A Firm-and Plant-Level Analysis of Production Sharing, Working Paper, Divisions of Research & Statistics and Monetary Affairs, Federal Reserve Board, Washington (D.C.), USA. http://www.federalreserve.gov/Pubs/FEDS/2006/200604/200604pap.pdf [01. Jan 2007].

Lee, J. A. N. (1992*a*), 'Claims to the Term "Time-Sharing"', *IEEE Annals of the History of Computing* **14**(1), S. 16–17.

Lee, J. A. N. (1992*b*), 'Time-Sharing at MIT: Introduction', *IEEE Annals of the History of Computing* **14**(1), S. 13–15.

Lehtisalo, K. (2005), *The History of NORDUnet – Twenty-Five Years of Networking Cooperation in the Nordic Countries*, NORDUnet. http://www.nordu.net/history/TheHistoryOfNordunet_simple.pdf [01. Jan 2007].

Licklider, J. C. R. (1960), 'Man-Computer Symbiosis', *IRE Transactions on Human Factors in Electronics* **HFE-1**, S. 4–11. http://groups.csail.mit.edu/medg/people/psz/Licklider.html [01. Jan 2007].

Licklider, J. C. R. und Vezza, A. (1978), 'Applications of Information Networks', *Proceedings of the IEEE* **66**(11), S. 1330–1346. http://ia311517.us.archive.org/0/items/ApplicationsOfInformationNetworks/AIN.txt [01. Jan 2007].

Litman, J. (2004), 'Sharing and Stealing', *Hastings Communications and Entertainment Law Journal* **27**. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=621261 [01. Jan 2007].

Lutterbeck, B. (2006), Die Zukunft der Wissensgesellschaft, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 445–465. http://www.opensourcejahrbuch.de/download/jb2006/chapter_09/osjb2006-09-03-lutterbeck [09. Jan 2007].

MCM (2006), 'The Pig and the Box'. http://www.dustrunners.com/Pig_and_the_Box.pdf [19. Jan 2007].

McCarthy, J. (1983), 'Reminiscences on the History of Time Sharing'. http://www-formal.stanford.edu/jmc/history/timesharing/timesharing.html [01. Jan 2007].

Meyers Lexikon online (2006), 'Timesharing'. http://lexikon.meyers.de/meyers/Timesharing [22. Jan 2007].

Parsons, R. D. (2002), 'Competition, Innovation, and Public Policy in the Digital Age: Is the Marketplace Working to Protect Digital Creative Works?', Anhörung. http://judiciary.senate.gov/hearing.cfm?id=197 [01. Jan 2007].

Pessach, G. (2007), 'An International-Comparative Perspective on Peer-to-Peer File-Sharing and Third Party Liability in Copyright Law: Framing the Past, Present, and Next Generation's Questions', *Vanderbilt Journal of Transnational Law* **40**. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=924527# [01. Jan 2007].

Rivoli, P. (2006), *Reisebericht eines T-Shirts. Ein Alltagsprodukt erklärt die Weltwirtschaft*, Econ, Berlin.

Strachey, C. (1959), Time Sharing in Large Fast Computers, *in* 'Proceedings of the International Conference on Information Processing', UNESCO, Paris, S. 336–341.

Strahilevitz, L. (2007), 'Wealth without Markets?', *The Yale Law Journal* **116**. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=946479 [07. Jan 2007].

Swire, P. P. (2006), 'Privacy and Information Sharing in the War on Terrorism', *Villanova Law Review* **51**(4), S. 951. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=899626 [01. Jan 2007].

Tushnet, R. (2006), 'TPRC on Peer Production: Opening Comments', 43(B)log. http://tushnet.blogspot.com/2006/09/tprc-on-peer-production-opening.html [19. Jan 2007].

U. S. Congress (2004), 'Intelligence Reform and Terrorism Prevention Act of 2004', Public Law 108–548. http://www.nctc.gov/docs/pl108_458.pdf [19. Jan 2007].

de Valk, G. und Martin, B. (2006), 'Publicly shared intelligence', *First Monday* **11**(9). http://firstmonday.org/issues/issue11_9/valk/index.html [01. Jan 2007].

van Schewick, B. (2004), Architecture and Innovation: The Role of the End-to-End Arguments in the Original Internet, Dissertation, Technische Universität Berlin.

Die Zukunft elektronischer Demokratie: Ideen, Methoden und Praktiken der Open-Source-Bewegung als Modell für die Softwareentwicklung in experimentellen Anwendungsfeldern

MAIKA BÜSCHENFELDT

(CC-Lizenz siehe Seite 563)

Wenn die zukünftige Form der Demokratie, wie es der amerikanische Demokratietheoretiker Robert A. Dahl ausdrückt, nicht im Vergangenen, sondern nur in einer schemenhaft erkennbaren Zukunft liegt, lassen sich die konkreten Ausformungen elektronischer Demokratie schwerlich in Prozessmodellen vorwegnehmen. E-Demokratie ist in diesem Sinne als experimentelles Anwendungsfeld zu begreifen, als ein Feld offener Sozialbezüge, in denen sich stabile Beteiligungsformen erst im Beteiligungsprozess herausbilden. Die Frage nach der Zukunft elektronischer Demokratie zielt damit auf die Offenheit für die Innovation sozialer Konstellationen und Praxisformen ab, auf einen evolutionär gedachten Prozess, der gerade durch seine Offenheit neue Praktiken und Verfahren der Partizipation hervorbringen kann. Diese Sichtweise erweitert das Verständnis elektronischer Demokratie in zweierlei Hinsicht: Zum einen werden nicht nur die Inhalte, sondern auch die Beteiligungspraktiken Gegenstand demokratischer Auseinandersetzung. Zum anderen werden die Bedeutung technischer Gestaltungsentscheidungen und damit auch Fragen der Softwareentwicklung thematisiert. Im Zentrum steht dabei die These, dass sich die demokratischen Potenziale des Internets erst in der ko-evolutionären Entwicklung von Software und Nutzungspraxis entfalten können.

Schlüsselwörter: E-Demokratie · Herrschaftsfreiheit · End-to-End-Prinzip · Ko-Evolution · Technik-Soziologie

1 Einleitung

> „Was die Zukunft betrifft, so ist es nicht unsere Aufgabe, sie vorauszusehen, sondern sie zu ermöglichen." (Antoine de Saint-Exupéry)

Auf der Website des Bundesinnenministeriums wird E-Demokratie als Sammelbegriff eingeführt, der alle Maßnahmen zusammenfasst, „bei denen Internettechnologien eingesetzt werden, um Bürgerinnen und Bürgern zusätzliche demokratische Mitbestimmungs- und Gestaltungsmöglichkeiten einzuräumen."[1] Diese Begriffsbestimmung lässt offen, mit welchen Praktiken, Ideen und Verfahrensweisen dieses „Mehr" an Partizipation verbunden ist. Es bleibt ferner offen, wie Bürgerpartizipation durch den Einsatz des Internets wirksam unterstützt werden kann. Diese Unbestimmtheit im Hinblick auf die konkreten Ausformungen demokratischer Beteiligungspraxis durchzieht die Begriffsklärungen der Abhandlungen und Studien zu diesem Thema und findet seinen Ausdruck darin, dass immer wieder das Fehlen konkreter Handlungsansätze und Erfahrungen bemängelt wird (z. B. bei Bräuer und Biewendt 2005 sowie Friedrichs et al. 2002 und Kollbeck et al. 2003).

Der amerikanische Demokratietheoretiker Robert A. Dahl sieht die Zukunft der Demokratie im Ungewissen:

> „Welche Form sie auch immer annehmen wird, die Demokratie künftiger Generationen will und kann nicht die Demokratie unserer Vorfahren sein. Die Grenzen und Möglichkeiten der Demokratie in einer Welt, deren Umrisse wir heute nur schemenhaft voraussehen können, werden sich sicherlich radikal von den Grenzen und Möglichkeiten früherer Demokratien unterscheiden." (Dahl 2002)

Wenn die zukünftige Form der Demokratie nicht im Vergangenen, sondern in einer nur schemenhaft erkennbaren Zukunft liegt, bleibt auch die konkrete Ausformung künftiger Partizipationspraktiken im Dunkeln und lässt sich schwerlich in Konzepten und Prozessmodellen vorwegnehmen. Die Unbestimmtheit bzw. das Fehlen konkreter Visionen der künftigen Beteiligungspraxis erscheint damit weder zufällig noch dramatisch, sondern eher als Ausdruck einer sich wandelnden Gesellschaft, die im Umgang mit neuen Kommunikationstechnologien und Partizipationspraktiken erst noch ihren Weg finden muss. Ich möchte E-Demokratie daher als ein experimentelles Anwendungsfeld einführen, als ein Feld offener Sozialbezüge, in denen sich noch keine stabilen Praxisformen etablieren konnten. *Experimentell* soll dabei heißen, dass dieses Anwendungsfeld gleichsam offen ist für Neues, für unvorhersehbare Entwicklungen und Strukturen, die sich aus der Praxis ergeben und nicht in Modellen vorweggenommen werden. Die Frage nach der Zukunft elektronischer Demokratie zielt damit auf

1 Diese Definition findet sich im „BMI – Lexikon der Innenpolitik" auf der Homepage des Bundesinnenministeriums: http://www.bmi.bund.de.

die Offenheit für die Innovation sozialer Konstellationen und Praxisformen ab, auf einen evolutionären Prozess, der gerade durch seine Offenheit neue Praktiken und Verfahren der Partizipation hervorbringt. Postuliert wird damit ein Demokratieverständnis, das auch die Ausgestaltung der Beteiligungsformen zu seinem Gegenstand macht.

Ausgehend von der Kennzeichnung elektronischer Demokratie als experimentelles Anwendungsfeld, wird in diesem Artikel die sozialwissenschaftliche Diskussion um die Potenziale elektronischer Demokratie aufgegriffen und um die Dimension technischer Gestaltungsentscheidungen erweitert. Ziel ist zunächst, die Bedeutung der Design-Entscheidungen von Software für die Ausgestaltung der konkreten Partizipationspraxis herauszustellen und die Entwicklung von Software als integralen Bestandteil einer E-Demokratie-Strategie zu verankern. Ausgehend von der These, dass sich die demokratischen Potenziale des Internets erst in der Verbindung von Softwareentwicklung und Nutzungspraxis entfalten können, werden anschließend Elemente eines Konzeptes für die Entwicklung von Software in experimentellen Anwendungsfeldern diskutiert. Im Mittelpunkt steht dabei die Softwareentwicklung in Open-Source-Communitys, die mit ihren Methoden und Ideen den Anforderungen eines ko-evolutionär ausgerichteten Softwareentwicklungsprozesses eher entgegenkommen als das klassische Softwareengineering.

2 Die Bedeutung von Softwarestrukturen in der E-Demokratie

Die Frage, welche Auswirkungen das Internet auf die Neugestaltung und Erweiterung direktdemokratischer Verfahren haben kann, beschäftigt die sozialwissenschaftliche Forschung, seitdem sich das Internet als massentaugliches Medium durchgesetzt hat. Bei Betrachtung des Forschungsstandes fällt zunächst auf, dass den vielen optimistischen Positionen nicht minder viele skeptische Beiträge entgegenstehen, während empirische Analysen nach wie vor vergleichsweise rar sind (Lührs et al. 2004; Schmidt et al. 2005).

Die Stärke des Internets wird in erster Linie darin gesehen, dass die Online–Kommunikation im Vergleich zu den klassischen Massenmedien nur mit minimalen Transaktionskosten verbunden ist und reziproke Kommunikationsformen zulässt. So kann im Internet prinzipiell jeder extrem kostengünstig und schnell publizieren.[2] Darüber hinaus erlauben die interaktiven Potenziale des Internets die kommunikative Vernetzung von Menschen. Howard Rheingold hat diese Kommunikationsstruktur als Many-to-Many-Kommunikation bezeichnet, als Kommunikation von jedem mit jedem (Rheingold 1993). Wird das Internet als modernes Massenmedium begriffen, ist in dieser Struktur verwirklicht, was bereits Berthold Brecht in seiner Radiotheorie forderte: Die Demokratisierung des Rundfunks durch seinen Wandel vom Distributions- zum Kommunikationsapperat, d. h. dem Vermögen, „den Zuhörer nicht nur

2 Vgl. dazu das *Online-Verwaltungslexikon*: http://www.olev.de.

hören, sondern auch sprechen zu machen und ihn nicht zu isolieren, sondern ihn in Beziehung zu setzen".[3] Ausdruck der großen Hoffnungen in die interaktiven Potenziale des Internets ist nicht zuletzt auch die Wiederbelebung der Diskussion um deliberative[4] Demokratiemodelle. Prominentes Beispiel ist Jürgen Habermas' Modell einer diskursiven Öffentlichkeit in Gestalt des rationalen, herrschaftsfreien Diskurses. In diesem Diskurs, der alle betroffenen Interessen berücksichtigt und nur das rationale Argument zum Zuge kommen lässt, sieht Habermas das wirksamste Mittel gegen die Demokratiedefizite repräsentativer Demokratien (Habermas 1996).

Im Zuge der Diskussion um die Digitale Spaltung der Gesellschaft hat sich jedoch auch eine kritische Gegenposition etabliert. Die Kritik am Internet als Vermittlungsmedium richtet sich dabei auf die hohen Anforderungen an Medienkompetenz, Technikkompetenz und Diskursfähigkeit, die letztendlich dazu führen würden, dass das Internet nicht die Inklusion aller relevanten Interessen befördere, sondern ganz im Gegenteil, den Ausschluss breiter Bevölkerungsteile (Hoecker 2002; Möller et al. 2006). Empirische Befunde scheinen diese Befürchtungen eher zu bestätigen als zu entkräften: So erscheint die Nutzung bzw. Nichtnutzung des Internets abhängig von Faktoren wie Geschlecht, Alter, formaler Bildung, Einkommen und Berufstätigkeit (vgl. dazu Hoecker 2002).

Neuere statistische Erhebungen zeigen zwar einen Trend der Nivellierung, z. B. in der Internetnutzung hinsichtlich des Geschlechts, stützen aber immer noch die These einer Digitalen Spaltung der Gesellschaft (Möller et al. 2006; Bautier und Demunter 2005). Am deutlichsten entzweien sich jedoch die Meinungen in der Beurteilung der Qualität von Nutzerbeiträgen. So schreibt Beate Hoecker, dass die in Internetforen geäußerten Beiträge wenig Anlass zur Hoffnung böten, dass die Diskursfähigkeit durch das Internet gestärkt würde (vgl. dazu Hoecker 2002). In der jüngeren Forschung hat sich zwar das Forschungsinteresse von den Internetforen auf die Weblogs verschoben, der zwischen Technikeuphorie und Kulturpessimismus schwankende Tenor ist jedoch geblieben. Weblogs gelten auch in der gegenwärtigen Weblogforschung ebenso als Verheißung verstärkter demokratischer Partizipation als auch als Ausgeburt der Banalität des Alltags, als „wieder ein Stück lauter gewordenes Medienrauschen" (Schmidt et al. 2005, S. 2).

Den euphorischen wie auch den skeptischen Positionen ist gemeinsam, dass die eingesetzte Software bzw. die Architektur des Internets in der Analyse weitestgehend unsichtbar bleibt. Software erscheint in der Regel als gegebenes, rein technisches und vermeintlich neutrales Produkt. Der Techniksoziologe Werner Rammert hat diese

3 Grundlage für Brechts Radiotheorie bieten die folgenden Schriften: *Der Rundfunk als Kommunikationsapparat. Rede über die Funktion des Rundfunks* (1932/33), *Vorschläge für den Intendanten des Rundfunks* (1928/29), der Aufsatz *Radio – eine vorsintflutliche Erfindung?* (1927/28) und das Hörstück *Der Flug der Lindberghs. Ein Radiolehrstück für Knaben und Mädchen* (1929).

4 Der Begriff Deliberation (engl.: die Beratung, die Überlegung oder die Bedächtigkeit) zielt auf öffentliche Kommunikation über politische Fragen auf Versammlungen, in Gremien und in der Medienöffentlichkeit (vgl. dazu http://de.wikipedia.org/wiki/Deliberation).

Sicht auf Technik für seine Disziplin als „Technikvergessenheit der Soziologie" kritisiert und eine interdisziplinäre Herangehensweise gefordert. Er richtet sich damit gegen die Trennung von Technik und Gesellschaft in einer Soziologie, die sich selbst behindert, indem sie die Fragen der Technikentwicklung und -gestaltung in die Umwelt der Gesellschaft verbannt (Rammert 1998).

In Anlehnung an Rammerts Programmatik, die Technik- und Gesellschaftsperspektive zusammenführt, möchte ich im Folgenden den Blick auf die technischen Architekturen des Internets und deren Gestaltungsoptionen lenken. Das führt zu einer Verschiebung der Perspektive und erweitert die Diskussion um die demokratischen Potenziale des Internets: Während sich der Fokus in der oben kurz skizzierten Kontroverse vorwiegend auf die reine Anwendung, d. h. die Nutzung und die Fähigkeiten bzw. Unfähigkeiten der Nutzer konzentriert, thematisiert die Erweiterung dieser Diskussion um Aspekte des Technischen den Zusammenhang von Nutzungspraxis und den Vorgaben, Beschränkungen und Potenzialen, die sich in den technischen Architekturen des Internets verbergen. Damit wird auch deutlich, was die Many-to-Many-Struktur des Internets voraussetzt, nämlich eine offene Infrastruktur als Voraussetzung für die Freiheitlichkeit des Internets. Es gerät auch in den Blick, was die Rede von der Many-to-Many-Struktur offen lässt, nämlich die Frage des *Wie*, d. h. welche Kommunikations- und Vernetzungspraxis in einer Many-to-Many-Struktur realisiert wird.

Es gibt einigen Anlass zu bezweifeln, dass das Internet unumstößlich freiheitlich gestaltet ist. So vertritt beispielsweise der amerikanische Rechtsgelehrte Lawrence Lessig die These, dass der Code das Gesetz des Cyberspace sei – *code is law*. Der Code, so seine These, entfalte als Regulationsinstanz mit ähnlicher Kraft wie Recht, Markt oder soziale Normen verhaltenssteuernde Wirkung. Die Architektur des Internets lässt sich in diesem Sinne als eine von Menschen geschaffene, sozial wirksame Umwelt begreifen, in der die in Code gefassten Architekturentscheidungen darüber bestimmen, welche Nutzungsweisen das Internet bietet und welche Nutzungsweisen es ausschließt (vgl. Lessig 2001). Die wichtigste Implikation dieser These im Hinblick auf die demokratischen Potenziale des Internets richtet sich damit auf die Freiheitlichkeit des Internets selbst.

Auch die Auseinandersetzung um die grundlegende Richtlinie für das technische Design des Internets, bekannt als die End-to-End-Richtlinie[5], ist Beispiel dafür, dass sich eine normative Debatte um Freiheit und Innovation mit technischen Argumenten führen lässt. In der End-to-End-Richtline wird festgelegt, dass ein Netzwerk lediglich für die Übermittlung der Daten zu sorgen habe, sodass an deren Enden eine beliebige Anwendung laufen kann. Tragendes Prinzip ist somit das Neutralitätsprinzip, d. h. der Verzicht auf jeglichen Ausschluss von Nutzern und Nutzungen (vgl. Lutterbeck

5 Das Konzept eines End-to-End-Designs ist 1984 zuerst in einem Papier von Jerome H. Saltzer, David P. Reed und David D. Clark vorgestellt worden. Der Titel des Papers: *End-to-end arguments in system design* (vgl. Saltzer et al. 1984).

2005, S. 340). In diesem Grundprinzip sehen viele Autoren die Grundlage für die Freiheit und die Offenheit des Internets für Innovationen. Informationstechnik trägt in dieser Argumentation als Infrastruktur wie Straßen, Schienenwege, das Telefon oder Elektrizität ihren Zweck nicht in sich selbst, sondern ermöglicht Innovation als Grundlage wohlstandsmehrenden Wirtschaftens (Lutterbeck 2005, S. 335).

Der Ruf nach Regulation und Änderungen im Netzdesign stellt das End-to-End–Prinzip und damit die Funktion des Internets als freie Infrastruktur jedoch in Frage. Christian Sandvig weist darauf hin, dass es technisch zwar durchaus zweckmäßig sein mag, das Netzwerk intelligenter zu machen und Funktionen wie Flusskontrolle, Caching oder Firewalls in das Netz zu integrieren, dass dies aber auch gleichzeitig die Möglichkeit eröffne, dass Netzbetreiber Kontrolle über die Enden ausüben (Sandvig 2003). Ob das Internet als neues Massenmedium seine Neuartigkeit in Gestalt der Many-to-Many-Kommunikation auch in Zukunft beibehalten wird oder diese der Regulation zum Opfer fällt, wird davon abhängen, wer die Kontrolle über die Enden ausübt und wie diese Kontrolle (aus-)genutzt wird.

Lawrence Lessigs These, dass der Code Gesetz sei, stellt jedoch nicht nur die Gefährdungen der Freiheit heraus, sondern hilft auch zu verstehen, dass die Struktur eines Softwaresystems die Form der Many-to-Many-Kommunikation beeinflusst. In der Software sind somit jene Regeln verborgen, die Einfluss darauf ausüben, wer mit wem auf welche Art und Weise kommunizieren kann. So sind Newsgroups, Chats, Diskussionsforen, Wikis oder Blogs Varianten einer Many-to-Many-Kommunikation, die eine jeweils unterschiedliche Beteiligungspraxis nahelegen.

Wenn die Struktur einer Software auch Einfluss auf die Nutzungspraxis hat, so ist dieser Zusammenhang aber auch nicht als deterministischer zu begreifen. Soziale Wirkungen lassen sich nicht ausschließlich aus den technischen Merkmalen einer Software ableiten. Die Softwarenutzung obliegt immer auch einem Prozess der sozialen Aneignung, in der ihre Verwendungsweisen sozial ausgehandelt und interpretiert werden (vgl. dazu Münch und Schmidt 2005).

Dass Software immer auch über Freiheitsgrade verfügt, die unterschiedliche Nutzungsweisen zulassen, zeigt sich zumeist darin, dass in der Nutzung von Software immer wieder Diskrepanzen zwischen den Erwartungen und der tatsächlichen Nutzungspraxis auftreten (vgl. Pape 2004). Zu diesen Diskrepanzen zählen u. a. Probleme des Missbrauchs, mangelnde Effizienz oder mangelnde Nutzung, aber auch die zum Teil überraschenden Wendungen in der Nutzungsgeschichte einer Anwendungssoftware. So haben sich beispielsweise Weblogs in den letzten Jahren vom zunächst sporadisch genutzten Online-Tagebuch zu einem bedeutsamen Instrument des Online–Journalismus und der Organisationsentwicklung entfaltet (vgl. Schmidt et al. 2005). Wenn die Struktur eines Softwaresystems die Beteiligungspraxis zwar wesentlich beeinflusst, aber auch nicht determiniert, lässt sich daraus folgern, dass auch im Feld elektronischer Demokratie der wechselseitigen Bedingtheit von Softwarenutzung und Softwarestruktur in angemessener Weise Rechnung getragen werden sollte.

3 Open Source als Modell für die Softwareentwicklung in experimentellen Anwendungsfeldern

In der Einleitung wurde das Feld elektronischer Demokratie als experimentelles Anwendungsfeld vorgestellt, d. h. als ein Anwendungsbereich von Software, der gerade durch Offenheit für unvorhersehbare Entwicklungen neuartige Praktiken und Verfahren der Partizipation hervorbringen kann. Der hier vorgestellte Ansatz verfolgt damit ein anderes Ziel als E-Demokratie-Projekte, die zwar auf die interaktiven Potenziale des Internets setzen, aber die soziale Praxis der Beteiligung durch ein Prozessmodell vorwegnehmen und die inhaltliche Auseinandersetzung in recht aufwändig strukturierten und moderierten Diskursen managen.[6]

In Abgrenzung zu diesem Vorgehen sollen im Folgenden Elemente eines Softwareentwicklungskonzeptes diskutiert werden, das die Partizipationspraxis nicht vorwegnimmt und davon ausgeht, dass sich Software nicht neutral verhält, sondern die Form der Partizipation mitbestimmt. Im Zentrum steht da die These, dass sich die demokratischen Potenziale des Internets erst in der ko-evolutionären Entwicklung von Software und Nutzungspraxis entfalten können. Ko-evolutionär soll hier heißen, dass Softwareentwicklung und Nutzungspraxis sich wechselseitig beeinflussen und gemeinsam neue Partizipationspraktiken hervorbringen. Die Entwicklung von Software wird damit – in Erweiterung der E-Demokratie-Diskussion – als integraler Bestandteil einer E-Demokratie-Strategie betrachtet.

Mit dieser Sichtweise ändern sich die Anforderungen an den Softwareentwicklungsprozess, weil Software in diesem Verständnis elektronischer Demokratie nicht zuviel vorwegnehmen darf und gleichsam offen sein muss für die evolutionäre Entwicklung neuer Praktiken. Dem traditionellen Softwareengineering sind hier Grenzen gesetzt, weil sich mit dem Fehlen etablierter Praktiken und vorgegebener Prozessmodelle wenig Anhaltspunkte für die Modellierung in einen auf Abbildung ausgerichteten und in Phasen organisierten Prozess der Softwareentwicklung finden lassen.

Die Entwicklungspraxis in Open-Source-Communitys hingegen scheint mit ihrem Paradigma der Offenheit andere Voraussetzungen für einen ko-evolutionär gedachten Entwicklungsprozess zu bieten. Open Source steht dabei zum einen für eine Tradition offener Standards und offener Quellcodes unter Verzicht auf die exklusiven Verwertungsrechte der entwickelten Software. Open Source steht zum anderen seit der Veröffentlichung von Eric Steven Raymonds Essay *The Cathedral and the Bazaar* aber auch für ein Softwareentwicklungsmodell, in dem Software in einer kooperativen Organisationsform hergestellt wird (Raymond 1999*a*).

6 Exemplarisches Beispiel ist das europäische DEMOS-Projekt, in dessen Rahmen von einem interdisziplinären Team eine Internetplattform zur politischen Bürgerbeteiligung entwickelt wurde. 2002 wurde auf dieser Plattform in Zusammenarbeit mit der Freien und Hansestadt Hamburg eine sehr umfangreiche Internetdiskussion zum Thema „Leitbild Metropole Hamburg – Wachsende Stadt" durchgeführt (vgl. dazu Lührs et al. 2004).

In diesem Entwicklungsmodell beschreibt Raymond den dominierenden Entwicklungsstil der Open-Source-Communitys als *Basarstil*, der mit frühen und häufigen Freigaben die Nutzer absichtsvoll möglichen Programmfehlern aussetzt und sich als ein „großer, wild durcheinander plappernder Basar von unterschiedlichen Zielsetzungen und Ansätzen" präsentiert (Raymond 1999*a*). Auch wenn die Bezeichnung Basar es nahelegt, so ist mit diesem Entwicklungsstil keinesfalls eine chaotische Projektorganisation zu verbinden. Open-Source-Projekte arbeiten zwar auf der Basis einer offenen und dezentralen Kooperation, sind dabei aber ausgesprochen wohlorganisiert und bedienen sich „strukturierter Prozesse mit klar definierten Rollen" (Bauer und Pizka 2005, S. 97). Mit dem Prinzip der offenen, kooperativen Projektorganisation bietet das Open-Source-Entwicklungsmodell grundsätzlich eine breite Basis für die Entwicklung von Software in experimentellen Anwendungsfeldern.

Anders als in der traditionellen Softwareentwicklung fehlen in der Entwicklung von Open-Source-Software die typischen Strukturierungen und Planungen des Gesamtprozesses, wie beispielsweise eine exakte Anforderungsanalyse (Bauer und Pizka 2005, S. 99). In experimentellen Anwendungsfeldern ist das jedoch kein Nachteil, sondern Vorteil, weil ohne den Zwang zu allzu strikter Strukturierung auch die Offenheit für Anpassungen und Weiterentwicklungen einer Software erhalten bleibt. Es wird damit möglich, auf die wechselnden Anforderungen zu reagieren, die sich aus der Nutzungspraxis ergeben.

Es ist somit gerade die vermeintliche „Planlosigkeit", die das Open-Source-Konzept für experimentelle Anwendungsfelder attraktiv macht. Hilfreich ist dabei vor allem, dass sich in der Open-Source-Entwicklung effiziente Alternativstrategien zur rigiden Projektplanung entwickelt haben. Die Entwicklung von Open-Source-Software hat Evolutionsstrategien hervorgebracht, die sich durch ausgesprochene Flexibilität auszeichnen. So wird das Design einer Software nicht im Voraus geplant, sondern entwickelt sich entsprechend der Dynamik wechselnder Anforderungen und der Projektgröße. Der steigenden Komplexität durch Wachstum der Codebasis und der Projektteilnehmer wird durch Modularisierung der Software begegnet (Bauer und Pizka 2005, S. 111).

Als weiterer Vorteil für die Softwareentwicklung in experimentellen Anwendungsfeldern kann sich die prinzipielle Unabgeschlossenheit des Open-Source-Modells erweisen. Im Hinblick auf die jüngeren Entwicklungstendenzen des Internets spricht Tim O'Reilly sogar von der Abschaffung des Software-Lebenszyklus (O'Reilly 2005). Hier wird einer der Grundgedanken der Open-Source-Bewegung wirksam, nämlich die Vorstellung, dass es sich bei Software nicht um ein Produkt, sondern um eine Dienstleistung handelt (O'Reilly 1999; Raymond 1999*b*). Diese Vorstellung findet ihre bislang ausgeprägteste Verkörperung in der Idee des Webservice, der grundsätzlich nicht verpackt und verkauft, sondern als Dienstleistung angeboten wird und das Internet als Plattform nutzt. Im Webservice tritt anstelle geplanter Releases die kontinuierliche Verbesserung der Software und anstelle der Lizenzierung oder des Verkaufs

die reine Nutzung (O'Reilly 2005). Als weitere Besonderheit der Open-Source-Entwicklungspraxis sticht das große Potenzial der Nutzerbeteiligung in der Entwicklung von Software hervor. Sowohl Raymonds Basarmodell als auch Tim O'Reillys Vision eines Web 2.0 setzen auf die Kraft des Sozialen – auf aktive Nutzer, die im Softwareentwicklungsprozess als Mitentwickler und Innovatoren involviert sind (Raymond 1999*a*; O'Reilly 2005).

Im erheblichen Potenzial, Nutzer zu Mitgestaltern und Innovatoren von Software zu machen, zeigt sich gleichsam Stärke und Schwäche der Open-Source-Entwicklung als Entwicklungsmodell experimenteller Anwendungsfelder. Gerade hier zeigt sich, dass das Zusammenwirken von Nutzung und Entwicklung einer Software auch in Open-Source-Projekten kein Automatismus ist. Problematisch bleibt, dass die Open–Source-Entwicklungspraxis in der Ausgrenzung und Nichtbeachtung von Nicht-Entwicklern ihre Schattenseiten zeigt. So bleibt häufig die Basis der vielbeschworenen Basisdemokratie, ebenso wie Offenheit, Selbstbestimmung und der „ideelle" Lohn der Anerkennung und des Ansehens auf die Gruppe der Entwickler beschränkt.

Die Meritokratien der Open-Source-Gemeinschaft sind bislang überwiegend Entwickler-Meritokratien geblieben, in welche die Beiträge nichtentwickelnder Community-Mitglieder, wie Dokumentation, *packaging*, Testen etc. tendenziell unterbewertet werden (vgl. dazu Jung 2006; Stürmer und Myrach 2006). Matthias Ettrich, der Begründer des KDE-Projekts, definiert Open-Source-Communitys entsprechend als „eine Gemeinschaft, basierend auf geteiltem Expertenwissen und gemeinsamen Interessen, in der Rang nur durch technisches Wissen und Taten erlangt werden kann" (Ettrich 2004). Die Gleichsetzung von Entwicklern und Nutzern stammt jedoch aus den frühen Tagen des Internets, als Entwickler noch für den Eigenbedarf und nicht für „Konsumenten" entwickelten (Grassmuck 2004).

Im Feld elektronischer Demokratie hingegen wird es auch darum gehen müssen, Nutzer nicht nur im Hinblick auf *usability* zu berücksichtigen, sondern auch an Design-Entscheidungen zu beteiligen. Sowohl bei Raymond als auch bei O'Reilly bleibt offen, wie Nicht-Entwickler in den Entwicklungsprozess einbezogen werden können. Strategien zu entwickeln, die auch die nichtentwickelnden Nutzer in die Entwicklung von Software einbeziehen, wird mit zu den größten Herausforderungen eines E-Demokratie-Ansatzes zählen, der den Entwicklungsprozess bewusst in sein Konzept einschließt.

Die These, dass sich die demokratischen Potenziale des Internets erst in der ko-evolutionären Entwicklung von Software und Nutzungspraxis entfalten können, ist empirisch nicht belegt. Dennoch, in der jüngeren Entwicklung des Internets häufen sich Beispiele, die den emanzipatorischen Gebrauch des Internets, auch jenseits normativ anspruchsvoller Deliberationsmodelle, deutlich werden lassen. Auf dem News-Portal *digg.com* (engl. *dig* = graben) beispielsweise, werden Beiträge nicht durch eine Redaktion ausgewählt, sondern durch die Nutzer. Dabei ist die grundlegende Funktion der Seite relativ einfach: Nutzer können die URL interessanter Beiträge auf die Platt-

form posten und diese durch die Vergabe so genannter *Diggs* durch andere Nutzer bewerten und ranken lassen. *Digg.com* ist damit ein Beispiel, wie der Wandel der massenmedial vermittelten Öffentlichkeit aussehen könnte, wenn über das Internet nicht nur Journalisten, sondern auch die Leser über die Relevanz von Themen und Nachrichten entscheiden (Agendasetting).

Ein weiteres Beispiel: Zur Fußball-Weltmeisterschaft 2006 hat die Netzzeitung mit der *ReadersEdition*[7] eine eigene Leserausgabe gestartet. Bereits in den ersten zwei Wochen haben mehr als 250 Autoren Texte zu Themen wie Politik, Wirtschaft, Sport oder Kultur geschrieben. Anders als in traditionellen Printmedien schreiben keine professionellen Journalisten, sondern die Leser selbst die Artikel. Der Platz, der ihnen dabei zu Verfügung steht, ist praktisch unbegrenzt. Begrenzung stellt nicht die Auswahl der Themen, sondern nur der Pressekodex dar, dessen Einhaltung durch eine Gruppe unabhängiger Moderatoren überprüft wird. Darüber hinaus gibt es keine Zensur.

Beide Beispiele reihen sich in eine Gruppe von Webanwendungen, die seit 2004 unter dem Begriff Web 2.0 zusammengefasst werden. Dieser Begriff ist das Ergebnis eines Brainstormings zwischen Tim O'Reilly und Dale Dougherty[8] und gilt als Oberbegriff für die Beschreibung einer Reihe neuer interaktiver Techniken und Dienste sowie der Beobachtung einer sich grundlegend verändernden Nutzungspraxis des Internets. In sozialer Hinsicht markiert der Begriff Web 2.0 den Wandel vom konsumierenden zum produzierenden Internetnutzer, der in selbst organisierten News-Portalen und auf Plattformen zum Austausch von Fotos, Videos, Radioepisoden zur Unterhaltung, Information, Aufklärung und Meinungsbildung beitragen kann. Kurz, im Web 2.0 zeichnet sich als Trend eine zunehmend emanzipatorische Nutzung des Internets ab, eine Dynamik von Offenheit und sozialer Innovation, vom Zusammenwirken von Entwicklung und Nutzung, die auch dem Feld elektronischer Demokratie neue Perspektiven eröffnen kann.

4 Fazit und Ausblick

In diesem Artikel wird das Verständnis elektronischer Demokratie in zweierlei Hinsicht erweitert: E-Demokratie wird zum einen als experimentelles Anwendungsfeld eingeführt, mit der Konsequenz, dass nicht nur die Inhalte, sondern auch die Struktur der Beteiligungspraxis selbst als Gegenstand demokratischer Auseinandersetzung betrachtet werden. Zum anderen wird die Bedeutung technischer Gestaltungsentscheidungen für die Ausformung der konkreten Partizipationspraxis herausgestellt. Mit dieser Erweiterung des E-Demokratie-Verständnisses stellen sich einige Fragen in der Auseinandersetzung um das demokratische Potenzial des Internets anders: So gerät in der Diskussion um den Ein- oder Ausschluss betroffener Interessen nicht

7 http://www.readers-edition.de

8 Vergleiche dazu die Homepage der deutschen Sektion des O'Reilly Verlags, http://www.oreilly.de/topics/web20_about.html.

nur das bestehende Webangebot und dessen Nutzung in den Blick, sondern auch die Anbieterseite. Dabei werden zwei Fragen virulent:

1. Von welchen *Voraussetzungen* ist der freie Zugang zum Internet abhängig, und zwar nicht nur in der Nutzung, sondern auch im Angebot von Inhalten und Webanwendungen?

2. Welche *Anforderungen* stellen sich an den Softwareentwicklungsprozess, wenn Software nicht als neutral begriffen wird und die Herausbildung der Partizipationspraxis nicht als planbar, sondern als offener, evolutionärer Prozess?

Vor dem Hintergrund dieser beiden Fragen kann Open Source in der Diskussion um die demokratischen Potenziale des Internets in zwei wichtigen Punkten Anregung bieten:

1. Wird Open Source als *Idee* begriffen, bietet sie mit ihrem zentralen Gedanken der Offenheit den Anstoß, über die Voraussetzungen von Vielfalt, Innovation und Freiheitlichkeit nachzudenken. So lässt sich Offenheit und der Verzicht auf Ausschluss als Voraussetzung der Many-to-Many-Struktur des Internets interpretieren, jener Kommunikationsstruktur also, in der das demokratische Potenzial des Internets im Wesentlichen verortet wird. Die End-to-End-Debatte mit ihrer Forderung nach einer offenen Infrastruktur mit offenen Standards berührt damit einen Aspekt, der in der Debatte der Digitalen Spaltung zumeist vernachlässigt wird, und zwar den Ausschluss von Interessen, der sich aus ungleichen Zugangschancen auf der Anbieterseite ergeben kann.

2. Wird Open Source als *Entwicklungsmodell* begriffen, bietet die Open-Source–Entwicklung mit ihrer besonderen Form der kooperativen Projektorganisation, ihrer Flexibilität und dem großen Potenzial aktiver Nutzerbeteiligung einen idealen Rahmen für die Entwicklung von Software in experimentellen Anwendungsfeldern.

Mit dem Fehlen etablierter Praktiken und vorgegebener Prozessmodelle sind dem traditionellen Softwareengineering Grenzen gesetzt, weil sich hier wenig Anhaltspunkte für die Modellierung in einem auf Abbildung ausgerichteten und auf in Phasen organisierten Prozess der Softwareentwicklung finden lassen. Die Entwickungspraxis der Open-Source-Gemeinschafthingegen scheint mit ihrem Paradigma der Offenheit, ihren spezifischen Strategien der Softwareevolution und dem großen Potenzial aktiver Nutzerbeteiligung wesentlich bessere Voraussetzungen für einen ko-evolutionär gedachten Entwicklungsprozess zu bieten.

Literatur

Bauer, A. und Pizka, M. (2005), Der Beitrag freier Software zur Software-Evolution, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 95–112. http://www.opensourcejahrbuch.de/download/jb2005/chapter_02/osjb2005-02-03-bauerpizka [09. Jan 2007].

Bautier, P. und Demunter, C. (2005), 'Die digitale Kluft in Europa: 85% der Schüler und Studenten nutzten 2004 das Internet', Eurostat Pressemitteilung 143/2005. http://epp.eurostat.cec.eu.int/pls/portal/docs/PAGE/PGP_PRD_CAT_PREREL/PGE_CAT_PREREL_YEAR_2005/PGE_CAT_PREREL_YEAR_2005_MONTH_11/4-10112005-DE-AP.PDF [09. Jan 2007].

Bräuer, M. und Biewendt, T. (2005), Elektronische Bürgerbeteiligung in deutschen Großstädten 2005: Zweites Website-Ranking der Initiative eParticipation, Studie, Initiative eParticipation, Berlin. http://www.initiative-eparticipation.de/Studie_eParticipation2005.pdf [09. Jan 2007].

Dahl, R. A. (2002), 'Robert A. Dahl zitiert nach: Moderner Staat – Moderne Verwaltung. Wegweiser Bürgergesellschaft', Bundesministerium des Inneren. http://www.bmi.bund.de/Internet/Content/Common/Anlagen/Broschueren/2002/Moderner__Staat__-__Moderne__Id__16553__de,templateId=raw,property=publicationFile.pdf/Moderner_Staat_-_Moderne_Id_16553_de.pdf [09. Jan 2007].

Ettrich, M. (2004), Koordination und Kommunikation in Open-Source-Projekten, *in* B. Lutterbeck und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2004 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 179–192. http://www.opensourcejahrbuch.de/download/jb2004/chapter_03/KoordinationUndKommunikation [09. Jan 2007].

Friedrichs, S., Hart, T. und Schmidt, O. (2002), 'Balanced E-Government. Visionen und Prozesse zwischen Bürgernähe und Verwaltungsmodernisierung', *Aus Politik und Zeitgeschichte* **52**(39-40), S. 12–23. http://www.bpb.de/publikationen/MKPWYU,0,0,%22Balanced_EGovernment%22:_Visionen_und_Prozesse_zwischen_B%FCrgern%E4he_und_Verwaltungs%3Cwbr%3Emodernisierung.html [09. Jan 2007].

Grassmuck, V. (2004), *Freie Software – Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn. http://freie-software.bpb.de/ [12. Dez 2006].

Habermas, J. (1996), *Die Einbeziehung des Anderen: Studien zur politischen Theorie*, Suhrkamp Taschenbuch Wissenschaft, Frankfurt am Main.

Hoecker, B. (2002), 'Mehr Demokratie via Internet? Die Potenziale der digitalen Technik auf dem empirischen Prüfstand', *Aus Politik und Zeitgeschichte* **52**(39-40), S. 37–45. http://www.bpb.de/publikationen/VNWAOH,0,0,Mehr_Demokratie_via_Internet.html [09. Jan 2007].

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

Gesellschaftsmodell', Lehmanns Media, Berlin, S. 235–250. http://www.opensourcejahrbuch.de/download/jb2006/chapter_05/osjb2006-05-03-jung [09. Jan 2007].

Kollbeck, J., Fazlik, N. und Ewert, B. (2003), E-Demokratie. Stand, Chancen und Risiken, *in* C. Schulzki-Haddouti (Hrsg.), 'Bürgerrechte im Netz', Bundeszentrale für politische Bildung, Bonn, S. 227–260. http://www.bpb.de/files/5XSXDC.pdf [09. Jan 2007].

Lessig, L. (2001), *Code und andere Gesetze des Cyberspace*, Berlin Verlag, Berlin.

Lutterbeck, B. (2005), Infrastrukturen der Allmende – Open Source, Innovation und die Zukunft des Internets, *in* B. Lutterbeck, R. A. Gehring und M. Bärwolff (Hrsg.), 'Open Source Jahrbuch 2005 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 329–346. http://www.opensourcejahrbuch.de/download/jb2005/chapter_05/osjb2005-05-05-lutterbeck [09. Jan 2007].

Lührs, R., Albrecht, S., Hohlberg, B. und Lübcke, M. (2004), 'Online Diskurse als Instrument politischer Partizipation – Evaluation der Hamburger Internetdiskussion zum Leitbild „Wachsende Stadt"', *kommunikation@gesellschaft* **5**. http://www.soz.uni-frankfurt.de/K.G/B1_2004_L%FChrs_Albrecht_Hohberg_L%FCbcke.pdf [09. Jan 2007].

Möller, E., Knippelmeyer, M. und Ahrens, K. (2006), (N)Onliner Atlas 2006. Eine Topographie des Digitalen Grabens durch Deutschland, Studie, TNS Infratest Holding GmbH & Co. KG and Initiative D21. http://www.nonliner-atlas.de/pdf/dl_NONLINER-Atlas2006.pdf [08. Jan 2007].

Münch, R. und Schmidt, J. (2005), Medien und sozialer Wandel, *in* M. Jäckel (Hrsg.), 'Mediensoziologie. Grundfragen und Forschungsfelder', VS Verlag für Sozialwissenschaften, Wiesbaden, S. 201–218. http://www.schmidtmitdete.de/pdf/MedienSozialerWandelPreprint.pdf [09. Jan 2007].

O'Reilly, T. (1999), 'Schlüsse aus der Open-Source-Software-Entwicklung', Telepolis. http://www.heise.de/tp/r4/artikel/6/6433/1.html [09. Jan 2007].

O'Reilly, T. (2005), 'What is Web 2.0. Design Patterns and Business Models for the Next Generation of Software'. http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html [09. Jan 2007].

Pape, B. (2004), Organisation der Softwarenutzung. Theoriebildung und Fallstudien zu Softwareeinführung und Benutzungsbetreuung, Dissertation, Universität Hamburg. http://asi-www.informatik.uni-hamburg.de/personen/pape/dissertation.html [09. Jan 2007].

Rammert, W. (1998), Technikvergessenheit der Soziologie? Eine Erinnerung als Einleitung, *in* W. Rammert (Hrsg.), 'Technik und Sozialtheorie', Campus, Frankfurt am Main, S. 9–28.

Raymond, E. S. (1999*a*), The Cathedral and the Bazaar, *in* 'The Cathedral and the Bazaar: Musings on Linux and Open Source from an Accidental Revolutionary', O'Reilly & Associates, Cambridge. http://www.catb.org/~esr/writings/cathedral-bazaar/ [09. Jan 2007].

Raymond, E. S. (1999*b*), 'Der verzauberte Kessel'. http://www.oreilly.de/opensource/magic-cauldron/cauldron.g.01.html [09. Jan 2007].

Rheingold, H. (1993), *The Virtual Community: Homesteading on the Electronic Frontier*, HarperPerennial. http://www.rheingold.com/vc/book/ [09. Jan 2007].

Saltzer, J. H., Reed, D. P. und Clark, D. D. (1984), 'End-To-End Arguments in System Design', *ACM Transactions on Computer Systems* **2**(4), S. 277–288. http://www.reed.com/Papers/EndtoEnd.html [09. Jan 2007].

Sandvig, C. (2003), Shaping Infrastructure and Innovation on the Internet: The End-to-End Network that isn't, *in* D. Guston und D. Sarewitz (Hrsg.), 'Shaping Science and Technology Policy: The Next Generation of Research', University of Wisconsin Press, Madison. http://www.spcomm.uiuc.edu/csandvig/research/Communication_Infrastructure_and_Innovation.pdf [09. Jan 2007].

Schmidt, J., Schönberger, K. und Stegbauer, C. (2005), 'Erkundungen von Weblog-Nutzungen. Anmerkungen zum Stand der Forschung', *kommunikation@gesellschaft* **6**. http://www.soz.uni-frankfurt.de/K.G/B4_2005_Schmidt_Schoenberger_Stegbauer.pdf [09. Jan 2007].

Stürmer, M. und Myrach, T. (2006), Open Source Community Building, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 219–234. http://www.opensourcejahrbuch.de/download/jb2006/chapter_05/osjb2006-05-02-stuermer-myrach [12. Dez 2006].

Produkthaftung für Open-Source-Software?

CHRISTIAN LAUX UND JAN WIDMER

(CC-Lizenz siehe Seite 563)

Wirtschaftlich hat sich Open Source heute etabliert: Viele Unternehmen haben ihr Geschäftsmodell auf Open Source ausgerichtet. Gerichtsentscheide zum Themenkomplex der Haftung und Gewährleistung für Open-Source–Software fehlen aber. Die juristische Fachschaft verbreitet die Warnung, dass das Haftungsrisiko für Open-Source-Software erheblich sei. Nicht selten wird behauptet, es sei für den Entwickler zu gefährlich, sich am Entwicklungsprozess von Open-Source-Software zu beteiligen. In erster Linie wird dabei die zwingende Produkthaftpflicht erwähnt. Der vorliegende Beitrag geht der Frage nach, ob jeder Entwickler, der Open-Source-Software auf eine öffentlich zugängliche Webseite hochlädt, unter dem Titel der Produkthaftung verantwortlich gemacht werden kann. Der Artikel zeigt auf, dass die Haftung nach den Produktehaftpflichtgesetzen von Deutschland und der Schweiz auf wichtige Vertriebsformen von Open-Source-Software nicht passt und dass die Haftungsfrage heute überbewertet werden dürfte.

Schlüsselwörter: Open-Source-Lizenz · aktiver Vertieb · passiver Vertrieb · Produkthaftung · Haftungsrisiko

1 Open-Source-Software

Open Source war in den letzten Jahren wirtschaftlich sehr erfolgreich. Die Geschäftsmodelle vieler Unternehmen basieren mitlerweile auf Open-Source-Software. Deshalb muss diese auch juristisch erklärt werden können. Eine wachsende Anzahl juristischer Publikationen befasst sich mit Open-Source-Software.[1] Während der große juristische Rahmen im Wesentlichen abgesteckt ist, ist sich die Lehre bezüglich wichtiger Aspekte nicht einig.

1 Vgl. insbesondere Jaeger und Metzger (2006), Spindler (2004) und Widmer (2003).

Dazu gehört auch das Thema der Haftung für Open-Source-Software. Da Gerichtsentscheide fehlen, bleiben die vorhandenen Stellungnahmen notgedrungen etwas theoretisch. Die juristische Fachschaft verbreitet die Warnung, dass das Haftungsrisiko für Open-Source-Software erheblich sei. In erster Linie wird die zwingende Produkthaftpflicht erwähnt (Jaeger und Metzger 2006, Rz. 225), ebenso eine weitreichende Produzentenhaftpflicht und gelegentlich sogar das Strafrecht.[2] Solche Stellungnahmen sorgen für große Verunsicherung unter Softwareentwicklern. Ist es für einen Programmierer zu riskant, Open-Source-Software zu entwickeln?

Man muss sich bewusst sein, dass rein rechtsdogmatische Stellungnahmen für gewöhnlich auch außerhalb der fachlichen Diskussion benutzt werden, um je nach Resultat *für* oder *gegen* Open Source Werbung zu machen. Es ist zu vermuten, dass nicht zuletzt aus diesem Grund das Thema rund um Haftung und Gewährleistung heute so stark umstritten ist. Die Reaktionen auf das Spindler-Gutachten (2003) haben dies veranschaulicht.[3]

Es würde zu viel Raum in Anspruch nehmen, die Grundlagen zu Gewährleistung und Haftung vollständig darzustellen. Der vorliegende Aufsatz beschränkt sich deshalb auf das Thema der Produkthaftung. Diese wurde geschaffen, um den Konsumenten vor den Gefahren von Konsumgütern zu schützen. Die Produkthaftung wurde rechtlich zwingend ausgestaltet und ist damit immun gegenüber Haftungsausschlussklauseln. Bedeutet das nun, dass jeder Entwickler, der Open-Source-Software auf eine öffentlich zugängliche Webseite hochlädt, unter dem Titel der Produkthaftung verantwortlich gemacht werden kann? Der vorliegende Aufsatz soll einen Denkanstoß zu dieser Frage geben. Da die Rechtslage nach deutschem sowie nach schweizerischem Recht weitgehend identisch ist, wird auf die anwendbaren Gesetzesbestimmungen beider Rechtsordnungen verwiesen.[4]

2 Rechtsbeziehungen beim Vertrieb von Open-Source-Software

Wenn Open-Source-Software weitergereicht wird, werden zeitgleich zwei unterschiedliche Rechtsverhältnisse begründet, die sogenannte Softwareüberlassung sowie eine Open-Source-Lizenz:

Softwareüberlassung Dem Empfänger wird die Software zugänglich gemacht. Denkbar

2 Paul (2005) erwähnt z. B. den Straftatbestand der Datenveränderung im Fall von Datenverlusten.

3 Das Gutachten Spindler hat für Aufruhr gesorgt. Der Auftraggeber der Studie, Der *Verband der deutschen Softwareindustrie* (*VSI*), interpretierte die Studie mit dem folgenden Kommentar: „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." Diese Meldung provozierte viele Reaktionen, vgl. etwa Knop (2003).

4 Zum deutschen Recht: Bürgerliches Gesetzbuch (BGB), Gesetz über die Haftung für fehlerhafte Produkte (ProdHaftG). Zum schweizerischen Recht: Obligationenrecht (OR, SR. 220), Bundesgesetz über die Produktehaftpflicht (PrHG, SR. 221.112.944).

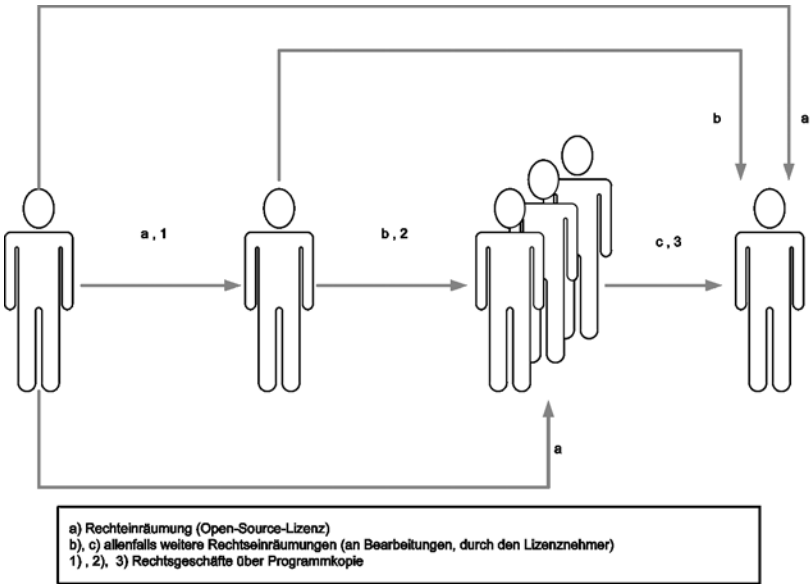


Abbildung 1: Rechtsverhältnisse bei Open-Source-Software

ist, dass ihm die Software auf einem Datenträger zugestellt wird, dass die Software in ein Hardware-Produkt integriert ist (sog. Embedded-Systeme) oder dass der Nutzer die Software per Download von der Webseite des Entwicklers oder eines Dritten[5] bezieht. Grundlage für die Softwareüberlassung ist ein Vertrag.

Open-Source-Lizenz Da das derart übermittelte Programm ohne eine Open-Source–Lizenz nur in engem Rahmen genutzt werden kann, bedarf es zur Ausübung der für Open-Source-Software typischen Freiheiten zusätzlich einer Lizenz, einer Open-Source-Software-Lizenz.[6]

Die erwähnte Zweiteilung wird in Abbildung 1 dargestellt. Die Rechtseinräumungen durch eine Open-Source-Lizenz werden mit Kleinbuchstaben bezeichnet, die Verträge zur Softwareüberlassung mit arabischen Ziffern.

5 Ein Beispiel wäre der Download von einer öffentlichen Plattform, wie beispielsweise *SourceForge* (http://sourceforge.net) oder *Savannah* (http://savannah.gnu.org).

6 Zur vertragsrechtlichen Einordnung auch des Lizenzverhältnisses vgl. das Urteil des Landgerichts Frankfurt a. M. vom 6. September 2006, Geschäfts-Nr. 2-6 0 224/06, S. 3, abrufbar unter http://www.jbb.de/urteil_lg_frankfurt_gpl.pdf; Entscheidung des Landgerichts München I vom 19. Mai 2004, Az 21 0 6123/04, S. 13, abrufbar unter http://www.jbb.de/urteil_lg_muenchen_gpl.pdf. Zur Diskussion in den USA siehe Madison (2003, 296 ff.) und Wacha (2005).

3 Bedeutung der Anonymität in der Vertragsbeziehung

Über das ihm übermittelte Softwarepaket steht der Empfänger, meist ohne dass er sich dessen bewusst ist, in mehreren Rechtsverhältnissen zu einer Vielzahl von Entwicklern. Für das Open-Source-Lizenzverhältnis sind sowohl die Unentgeltlichkeit als auch die Anonymität der Vertragsbeziehung typisch.[7] Für die Softwareüberlassung ist zwischen *passivem* und *aktivem* Vertrieb zu differenzieren.

3.1 Passiver Vertrieb

Die Softwareüberlassung ist oft ebenfalls anonym. Dies ist dann der Fall, wenn die Software von einer öffentlichen Plattform heruntergeladen wird, sei es von derjenigen des Entwicklers, sei es von jener eines Dritten. Typischerweise erfolgt der Download von diesen Seiten, ohne dass der Entwickler dies bemerkt. Der Download erfolgt auf Initiative des Empfängers der Software. Diese Form des Vertriebs von Open–Source-Software kann als *passiver Vertrieb*[8] bezeichnet werden. An der Qualifikation als passiver Vertrieb ändert sich auch dann nichts, wenn ein Komitee über die Einbindung von Programmkomponenten entscheidet;[9] maßgeblich ist die Form der Verbreitung.

Wer Open-Source-Software entgegennimmt, wird allein durch diese Tatsache nicht verpflichtet, gleichwohl liegt der Softwareüberlassung ein Vertragsverhältnis zu Grunde. Die rechtliche Würdigung dieser Vertragsbeziehung ist in der Lehre heftig umstritten.[10] Darauf muss vorliegend jedoch nicht weiter eingegangen werden, da die rechtliche Qualifikation der Vertragsbeziehung die produkthaftungsrechtliche Diskussion nicht beeinflusst.

---

7 Oft ergibt sich die Nutzungsberechtigung des Empfängers nicht aus dem Lizenzvertrag, sondern bereits aus dem Gesetz (vgl. Art. 12 Abs. 2 URG; Art. 17 Abs. 1 URV; § 69d Abs. 1 UrhG; Art. 5 Nr. 1 der EG-Richtlinie 91/250/EWG vom 14. Mai 1991 über den Rechtsschutz von Computerprogrammen). Erst wenn der Empfänger Nutzungshandlungen vornimmt, die ohne Lizenz verboten wären, kann angenommen werden, dass zusätzlich zum Softwareüberlassungsvertrag auch ein Lizenzvertrag zu Stande kommt. Hierzu siehe Hoeren (2004) und Jaeger und Metzger (2006, Rz. 204).

8 In der Lehre wird oft der Begriff des *isolierten Vertriebs* verwendet (vgl. etwa Spindler 2004); generell Taeger (1995, S. 211). Da dieser Begriff eher die Trennung des Programms von anderen Programmkomponenten als die Unentgeltlichkeit und Anonymität der Vertragsbeziehung – worauf es ankommt – unterstreicht, wird hier zur Abgrenzung ein anderer Begriff geprägt.

9 Dies ist von Bedeutung im Zusammenhang mit der Haftungsexposition des Komitees (vgl. Spindler 2004). Zu erwähnen ist, dass die meisten Projektleitungen heute in Form einer Stiftung, zum Beispiel http://www.eclipse.org/org, oder einer Gesellschaft, vgl. http://www.mozilla.org/foundation organisiert sind.

10 Zur Diskussion, ob die Softwareüberlassung als Schenkung zu behandeln ist vgl. die folgende Literatur: Jaeger und Metzger (2006, Rz. 206, Fn. 567–570) mit Hinweisen auf Judikatur und Lehre; Deike (2003, S. 14 f.); Schiffner (2003, S. 255) – beschränkt auf körperlichen Übertragungsvorgang, für Download: Gefälligkeitsverhältnis; Spindler (2004); Koch (2000, S. 335); Lenhard (2006, S. 376); Weber (2002*b*, S. 57); Weber (2003*b*). Gegen Schenkungsrecht: Hoeren (2004, S. 238) und Sester (2000, S. 799 ff.).

3.2 Aktiver Vertrieb

Wird Open-Source-Software demgegenüber von einem spezialisierten Dienstleister (z. B. *RedHat* oder *Novell*) vertrieben oder hergestellt, wird die Anonymität überwunden. Der Anbieter der Software wird den Empfänger kennen oder zumindest nach geringerem Aufwand namentlich bestimmen können. Zur Abgrenzung kann man hier von *aktivem Vertrieb* sprechen. Im aktiven Vertrieb sind die Leistungen des Anbieters in der Regel entgeltlich.

4 Allgemeines zur Produkthaftung für Open Source

Aus Sicht des Verbraucherschutzes besteht das berechtigte Anliegen, den modernen Industriegefahren flächendeckend zu begegnen. Diese Gefahren können auch durch Software hervorgerufen werden, was heute keiner besonderen Erläuterung mehr bedarf.

4.1 Grundlagen

Es besteht vermehrt die Tendenz, Software als ein Produkt im Sinne der Produkthaftpflichtgesetze (in der Schweiz: PrHG; in Deutschland: ProdHaftG) zu begreifen, obwohl Software in den maßgeblichen Gesetzesbestimmungen nicht ausdrücklich als Gegenstand der Produkthaftpflicht genannt wird.[11] Diese Tendenz wird im vorliegenden Beitrag nicht hinterfragt, obwohl die Diskussion noch im Gang ist. Nach dem Produkthaftungsrecht haftet der Hersteller eines Produkts für Körperschäden (Art. 1 Abs. 1 lit. a PrHG; § 1 Abs. 1 ProdHaftG) und auch für Sachschäden, sofern sie nicht im gewerblichen Bereich des Geschädigten eintreten (Art. 1 Abs. 1 lit. b PrHG; § 1 Abs. 1 ProdHaftG). Wenn in der Lehre Gesundheitsschädigungen mit hohen Haftungsobergrenzen erwähnt werden, ist dies nicht falsch. Damit wird aber eher das theoretisch mögliche Risiko und nicht die wahrscheinlichste Risikoexposition beschrieben.[12] Hauptanwendungsfall dürfte vielmehr der Datenverlust bzw. die Daten-

11 Das gilt für proprietäre Programme gleich wie für Open-Source-Software. Zur Rechtslage in der Schweiz: Straub (2002, Rz. 13 ff.). Zur Rechtslage in Deutschland: Taeger (1995, S. 160 ff.); Spindler (2004, S. 195); Jaeger und Metzger (2006, Rz. 227).

12 Wenn ein Hersteller eines medizinischen Steuergeräts vorbestehende Open-Source-Software einsetzt, so haftet jedenfalls er – sowohl aus Vertrag als auch aus Produkthaftpflicht – für durch Fehlfunktionen entstandenen Schaden. Das Spital wird zusammen mit den Geräten (samt Software) finanzielle Sicherheiten verlangen. Es ist deshalb unvorstellbar, dass das Spital im Schadensfall auch nur entfernt daran denken würde, auf jemand andern als den Vertragspartner (bzw. die von ihm gestellte Sicherheit) zuzugreifen. Wie dem auch sei: Die meisten Open-Source-Lösungen für diesen Bereich beschränken sich heute soweit ersichtlich nach wie vor auf Programme zur Verwaltung von Labor- und Patientendaten (vgl. http://resmedicinae.sourceforge.net/analysis/littlefish/links/fishlinks.htm; http://librehealth.sourceforge.net/inventory.html), gelegentlich finden sich Anwendungen mit erweiterten Funktionen (z. B. für Telepathologie/-medizin, http://ipath.sourceforge.net/).

veränderung auf der Festplatte des privat genutzten Computers sein.[13] Ob Datenverlust oder Datenänderung jedoch als Sachschaden im Sinne des Produkthaftungsrechts gelten kann, wird kontrovers diskutiert.[14] Es ist also denkbar, dass das Produkthaftungsrecht auf die in der Praxis häufigste Fallkonstellation gar nicht anwendbar ist.

4.2 Herrschende Lehre

In der herrschenden Lehre wird die Produkthaftpflicht auch für Open-Source-Software grundsätzlich bejaht (siehe Spindler (2004, S. 194 ff.), Jaeger und Metzger (2006, Rz. 225 ff.) und Schiffner (2003, S. 258 ff.)). Abgelehnt wird sie nur dort, wo der Entwickler das Programm in der Freizeit geschrieben hat und Hinweise auf eine besondere berufliche Qualifikation des Entwicklers fehlen (Spindler 2004, S. 202; Jaeger und Metzger 2006, Rz. 229). Würde die strikte Produkthaftpflicht aber – mit der herrschenden Lehre – abgesehen von wenigen Ausnahmen immer auch für Open-Source–Software greifen, so würde dies für die vielen Anbieter, die Software im passiven Vertrieb anbieten, ein nicht kontrollierbares Risiko darstellen.[15] Im Falle der Fehlerhaftigkeit von Open-Source-Software könnte ein jeder Entwickler haftbar gemacht werden; denn die Haftungsausschlüsse in den Open-Source-Software-Lizenzen können den Produkthaftungsanspruch nicht einschränken oder beseitigen (Art. 8 PrHG und § 14 ProdHaftG; vgl. z. B. Zirlick (2003, S. 202)). Dieses Resultat weckt Bedenken, auf die der folgende Abschnitt näher eingeht.

4.3 Bedenken

Die Produkthaftpflicht passt nicht zum passiven Vertrieb.[16] Dies wird ersichtlich, wenn die zur Rechtfertigung der Produkthaftpflicht vorgetragenen Argumente daraufhin untersucht werden, ob sie auch für Open-Source-Software im passiven Vertrieb herangezogen werden können.

Risikoallokation Unter diesem Stichwort wird angeführt, der Softwareentwickler als *cheapest cost avoider* (vgl. Morscher (1999, S. 66) und Ann (1991, S. 115)) könne den Schaden am besten vermeiden, also habe er ihn auch zu tragen. Dieses Argument überzeugt im passiven Vertrieb jedoch gerade nicht. Der Entwickler, der seinen Quelltext auf ein öffentliches Repository lädt, kann nicht voraussehen, wozu der Nutzer die Software verwenden wird. Allokationsüberlegungen

13 Dies erklärt, weswegen Gerichtsentscheide zur Produkthaftung bislang fehlen. Schäden an privat genutzten Gegenständen haben meist einen zu geringen Wert, als dass der Geschädigte den Aufwand und das Prozessrisiko eines Gerichtsverfahrens auf sich nehmen würde.

14 Vgl. zum Stand der Lehre, ob Datenverlust als Sachschaden qualifiziert: Bartsch (1998), Günther (2001), Schiffner (2003), Straub (2004), Weber (2003*a*).

15 Spindler (2004, S. 203) weist zu Recht darauf hin, dass für solches Tätigwerden heute kein Versicherungsschutz erhältlich scheint.

16 Vgl. zur hier verwendeten Begriffsbestimmung Abschnitt 3.1

müssen bei Open-Source-Software vielmehr dazu führen, dass der Entwickler (bei passivem Vertrieb) grundsätzlich nicht haftet.

Voraussehbarkeit des Schadenseintritts Hierzu wird Folgendes angeführt:

> „Der Schaden sollte nicht demjenigen verbleiben, der mehr oder weniger zufällig von ihm betroffen wird, sondern von demjenigen getragen werden, der das Schadensrisiko wenigstens im allgemeinen beherrscht." (Taeger 1996, S. 270)

Sofern der Geschädigte die schädigende Software nicht selber heruntergeladen hat, mag zutreffend sein, dass er den Schadenseintritt nicht voraussehen konnte. Dieses Argument führt jedoch nicht dazu, dass der Entwickler im passiven Vertrieb zur Rechenschaft zu ziehen ist. Der Entwickler gibt die weitere Verwendung der Software aus der Hand und kann sie nicht kontrollieren (Lutterbeck et al. 2000, S. 110). Ein allfälliger Schadenseintritt ist für ihn in keiner Weise voraussehbar. Eine Software kann mit Blick auf die eine Verwendungsart fehlerfrei sein, während sie für eine andere untauglich ist. Das Beispiel des Ariane-5-Desasters verdeutlicht dies: Das Programm, welches schließlich zur Zerstörung der *Ariane 5* führte, funktionierte jahrelang reibungslos – in der Vorgängerrakete *Ariane 4*. Das Programm war aber für die *Ariane 5* wiederverwendet worden und verursachte dort falsche Resultate (vgl. Lions 1996). Das Testen eines Programms auf seine Funktionsfähigkeit hin setzt demnach voraus, dass die Spezifikation für den konkreten Einsatzbereich bekannt ist. Das ist im passiven Vertrieb von Open-Source-Software regelmäßig nicht der Fall. Der Entwickler von Open-Source-Software sollte deshalb nicht haften, wenn er die Software nur passiv anbietet.

Nutzen Unter diesem Aspekt wird angeführt, der Schaden solle von demjenigen getragen werden, der den wirtschaftlichen Nutzen aus dem Gebrauch eines Wirtschaftsguts hat. Die Tatsache, dass der Empfänger die Software gebraucht, nützt nun aber gerade beim passiven Vertrieb nur dem Nutzer, nicht aber dem Entwickler; denn der Entwickler erhält keine Gegenleistung für die Softwareüberlassung.[17] Auch diese Überlegungen führen dazu, die Produkthaftung des Entwicklers von Open-Source-Software zu verneinen, wenn der Entwickler die Software nur passiv anbietet.

Versicherungsmöglichkeit Die meisten privaten Haftpflichtversicherungen decken das aus dem Upload von Software auf das Internet entstehende Schadensersatzrisiko in der Regel nicht ab.[18] Entsprechend geht das zur Begründung der Produkt-

17 Vgl. Abschnitt 3.1.

18 Siehe Fußnote 15.

haftpflicht jeweils angeführte Argument, wonach der Entwickler das Risiko auf eine Versicherung abwälzen könne, ins Leere.

Transaktionskosten Unter diesem Argument wird im Allgemeinen versucht, die Entscheidung eines kommerziellen Softwareanbieters mit traditioneller wirtschaftlicher Logik nachzuzeichnen. Übersteigen die Kosten der Produktion (samt Versicherung) die erwarteten Einnahmen, so werde der Entwickler von der – wirtschaftlich demzufolge ineffizienten – Softwareentwicklung absehen. Auch diese Überlegung ist für Open-Source-Software im passiven Vertrieb nicht angebracht, denn sie lässt die Motivationen zur Herstellung von Open-Source-Software außer Acht. Dieser Begründungsansatz würde bei Open-Source-Software in allen denkbaren Konstellationen zum Schluss führen, dass es Open-Source–Software nicht geben dürfe, denn die Bejahung der Produkthaftung könnte viele Entwickler davon abhalten, sich weiter an Open-Source-Entwicklungsprojekten zu beteiligen. Das genannte Argument führt zu nicht haltbaren Resultaten.

Die Analyse der traditionellen Begründungen für das Produkthaftpflichtrecht lässt nur den Schluss zu, dass das Produkthaftpflichtrecht für Open-Source-Software (im passiven Vertrieb) nicht das richtige Instrument ist.[19] Nach dem Produkthaftungsrecht gilt der Umstand als haftungsbegründend, dass ein Produkt in Verkehr gebracht wurde, obwohl es für den vorgesehenen Zweck nicht tauglich war:

> „Produkthaftung setzt ein im Inverkehrbringen eines fehlerhaften Produkts liegendes rechtswidriges Verhalten [. . .] voraus." (Taeger 1995, S. 102)

Für den passiven Vertrieb ist nun entscheidend, dass der Entwickler ganz augenscheinlich nicht in der Lage ist, das Programm für den schließlich verwendeten Einsatzzweck tauglich zu machen – und zwar mangels Voraussehbarkeit. Der Nutzer des Programms hätte umgekehrt aber alle notwendigen Instrumente hierzu in der Hand (insbesondere Einsicht in den Quelltext). Deshalb darf allein die Tatsache, dass Open-Source-Software auf einer öffentlich zugänglichen Plattform zum Download angeboten wird, keine Produkthaftung auslösen.

Es ist hier nicht der Ort, die Grundlagen der Produkthaftpflicht dogmatisch lückenlos aufzubereiten. Die nachstehende Darstellung beschränkt sich deshalb darauf, die wichtigsten Argumente darzustellen, die aus Entwicklersicht gegen die Anwendbarkeit des Produkthaftpflichtrechts angeführt werden können.

19 Spindler (2004, S. 203) räumt zwar ein, dass die Produkthaftpflicht im Einzelfall durchaus unpassend sein könne, ist aber der Auffassung, dass Verallgemeinerungen nicht möglich seien. Vielmehr müsse *im Zweifel* davon ausgegangen werden, dass ein wirtschaftlicher Zweck vorliege und damit das Produkthaftpflichtrecht zur Anwendung komme.

5 Keine Produkthaftpflicht im passiven Vertrieb

Nachstehend wird dargelegt, welche Abwehrargumente dem Entwickler zur Verfügung stehen, um die unerwünschten Nachteile abzuwenden, die sich bei Anwendbarkeit des Produkthaftpflichtrechts auf Open-Source-Software ergeben würden. Die Abwehrargumente beruhen im Wesentlichen auf dem Umstand, dass für den Entwickler von Open-Source-Software in der Regel nicht voraussehbar ist, wie sein Entwicklungsbeitrag schließlich verwendet wird.[20] Es kann differenziert werden danach, ob der Geschädigte die Software direkt von der Webseite des Entwicklers bezogen hat (dazu Abschnitt 5.2) oder nicht (dazu Abschnitt 5.1).

5.1 Schädigung eines Dritten ohne Vertragsverhältnis

Steht der Geschädigte in keinem Vertragsverhältnis zum Entwickler, kann der Entwickler gegen die Produkthaftung insbesondere die folgenden Argumente anführen:

Information objektiv ungefährlich Zumindest, wenn die Software nur im Quelltext vorliegt, kann mit guten Gründen gegen die Anwendung des Produkthaftungsrechts argumentiert werden. Quelltext ist noch viel stärker als kompilierte Software bloße Information (Weber 2002*a*, S. 1016). Bloße Informationen sollten keine Produkthaftung auslösen, solange sie nicht in ein Produkt mit Schadenspotenzial eingebunden worden sind. Oft kann der Quelltext ohne vorgängige Kompilation gar nicht dazu eingesetzt werden, eine Datenverarbeitungsanlage anzutreiben.[21] Trifft dies zu, ist die im Quelltext verkörperte Information objektiv ungefährlich.[22] Wer bloße Information zur Verfügung stellt, sollte deshalb noch nicht als Hersteller eines Teilprodukts behandelt werden (vgl. Günther 2001, S. 241 mit Verweisen). Einschränkend ist Folgendes zu erwähnen: Da Computerprogramme heute immer häufiger direkt nur im Quellcode installiert werden und die Maschine den Code erst im Runtime-Modus kompiliert, dürften die Anwendungsfälle für dieses Argument bei realistischer Betrachtung immer stärker zurückgehen.

Fehlende Produzenteneigenschaft Open-Source-Software befindet sich in einem laufenden Entwicklungsprozess, dessen Abschluss nicht vorausgesagt werden kann. Dies ist ein wesentlicher Unterschied zur proprietären Software, bei der es im Belieben des Softwareunternehmes liegt, den Abschluss des Entwicklungsprozesses und damit die Marktreife zu bestimmen. Hat ein Dritter die auf der Entwicklerplattform verfügbare Software heruntergeladen und anschließend bis zur Marktreife gebracht, muss dieser als Produzent gelten, nicht aber der Entwickler der *Rohversion*. Dies muss jedenfalls dann gelten, wenn ein anderer als der

20 Siehe Abschnitt 4.3.

21 Siehe Straub (2004, S. 261) und Straub (2002, Rz. 28).

22 Zur Haftung für reine Information vgl. Straub (2002, Rz. 17 ff.), Hilty (2001, S. 813 ff.).

ursprüngliche Entwickler die Software zur Marktreife gebracht hat. Ein Dritter, meist ein Mehrwertdienstleister, hat die Software übernommen und für einen bestimmten Zweck tauglich gemacht. Wenn der ursprüngliche Entwickler den Einsatzbereich der Software nicht kannte, hat der spätere Entwickler den ursprünglichen faktisch vollständig substituiert. Mit anderen Worten ist der nachfolgende Mehrwertdienstleister an die Stelle des ursprünglichen Entwicklers getreten und hat diesen ersetzt. Der Entwickler solcher isoliert vertriebener Open-Source-Software kann unter diesen oft vorliegenden Umständen nicht als Produzent gelten. Mit Blick auf den heutigen Stand in der Literatur ist dieses Argument allerdings problematisch. Auch ein Entwickler von Bestandteilen des schließlich fehlerhaften Produkts gilt nach dem Produkthaftpflichtrecht üblicherweise als Produzent (vgl. Fellmann (1996, N 6 f.) zu Art. 2 PrHG; Straub (2002, Rz. 71 ff.)), eine Position, die in Bezug auf Open-Source-Software jedenfalls überdacht werden sollte.

Nicht in Verkehr gebracht[23] In den Verkehr wird die Software dann gegeben, wenn sie mit dem Ziel der Vermarktung aus dem eigenen Herrschaftsbereich herausgegeben wird (siehe Taeger (1996, S. 265); Straub (2002, Rz. 132 ff.); Fellmann (1996, N 2 f.) zu Art. 5 PrHG). Mit dem Abrufbarmachen auf einer öffentlich zugänglichen Webseite hat der Entwickler die Software zweifelsohne aus seinem Herrschaftsbereich entlassen. Ob dies zur Vermarktung gegenüber einem Konsumenten erfolgte, ist abhängig davon, ob die Software *tel quel* vom Konsumenten verwendet werden kann oder ob das Programm erst noch in ein weiteres Programm eingebunden werden muss, um eingesetzt werden zu können.[24] Sind weitere Entwicklungsarbeiten oder auch nur die Verknüpfung mit einem Programm erforderlich, kann wiederum mit guten Gründen gegen die Anwendbarkeit der Produkthaftung argumentiert werden.[25]

Fehlende Voraussehbarkeit Wenn Open-Source-Software passiv vertrieben wird, kann jedermann die Software herunterladen. Der Entwickler kann somit in der Regel nicht voraussehen, auf welchen Plattformen und in welchem Installationsumfeld seine Software eingesetzt werden wird; die im Zusammenhang mit der Software möglicherweise entstehenden Risiken sind für den Entwickler deshalb nicht im Voraus erkennbar. Umgekehrt hat es der Empfänger der Software in der Hand, die Risiken des Softwareeinsatzes zu kontrollieren, denn er hat im Gegensatz zu proprietärer Software die Möglichkeit, den Quelltext selber nach Sicherheitsdefiziten zu überprüfen.[26] Da der Entwickler ganz anders als

23 Nach Art. 5 Abs. 1 lit. a PrHG; § 1 Abs. 2 Ziff. 1 ProdHaftG.

24 Was zum Beispiel bei Programmbibliotheken der Fall ist.

25 Siehe zur fehlenden Voraussehbarkeit bereits Abschnitt 4.3.

26 Vgl. – allerdings zur Produzentenhaftung/Deliktshaftung – Spindler (2004, S. 193); vgl. auch BGH vom 5. Mai 1992: VI ZR 188/91, NJW 1992, 2016, 2018; BGH vom 14. Mai 1996: VI ZR NJW 1996, 2224, 2226.

der Empfänger der Software die Art der Verwendung seiner Software nicht voraussehen kann, darf der Fehlerbegriff für passiv vertriebene Open-Source–Software nur in Ausnahmefällen bejaht werden. Ein solcher Ausnahmefall kann angenommen werden, wenn die Software für einen ganz bestimmten Einsatz tauglich gemacht wurde und sich der Nutzer auf die Tauglichkeit zu diesem Zweck verlassen durfte, z. B. wenn die Open-Source-Software als selbständig lauffähiges *executable* verfügbar gemacht wird und es abgesehen von der Installation durch den Nutzer keiner weiteren Schritte bedarf, damit die Software auf einem bestimmten System einsetzbar ist (§ 3 Abs. 1 lit. b ProdHaftG).[27] Produkthaftpflicht kann sich auch daraus ergeben, dass der Entwickler der Open-Source-Software seine Software besonders anpreist und so besondere Sicherheitserwartungen weckt, z. B. mit Mitteilungen im Umfeld der Download–Möglichkeit (vgl. Art. 4 Abs. 1 lit. a PrHG; § 3 Abs. 1 lit. a ProdHaftG). Fehlt es an solchen qualifizierenden Umständen, muss angenommen werden, dass eine Fehlfunktion und somit ein Fehler nicht so sehr mit der Open-Source-Software selber verknüpft ist, sondern vielmehr deshalb entstanden ist, weil die Software in einer für den Entwickler nicht voraussehbaren Art und Weise eingesetzt wurde. Sollte ein Fehler gleichwohl bejaht werden, sind jedoch die Haftungsausschlussgründe gemäß § 1 Abs. 2 Ziff. 2 ProdHaftG bzw. Art. 5 Abs. 1 lit. b PrHG heranzuziehen. Nach diesen Bestimmungen ist die „Ersatzpflicht des Herstellers [. . .] ausgeschlossen, wenn nach den Umständen davon auszugehen ist, daß das Produkt den Fehler, der den Schaden verursacht hat, noch nicht hatte, als der Hersteller es in den Verkehr brachte" (§ 1 Abs. 2 Ziff. 2 ProdHaftG). Der Entwickler ist beweispflichtig, dass dieser Haftungsausschlussgrund zur Anwendung kommt (Fellmann 1996, N 1 zu Art. 5 PrHG). Nach Art. 5 Abs. 1 PrHG muss der Entwickler zwar den Beweis der Fehlerlosigkeit antreten. Unseres Erachtens muss dieser Beweis aber bereits dann gelingen, wenn der Entwickler nachweist, dass der Fehler teilweise auf mangelnde Kompatibilität mit dem System zurückzuführen ist, auf welchem die Software zum Einsatz kam. Ist der Fehler auf mangelnde Kompatibilität der Software mit anderen Programmen oder der Systemumgebung zurückzuführen, liegen Umstände vor, die zeigen, dass der Fehler noch nicht vorlag, als das Programm in Verkehr gebracht wurde.

27 Eine bloße Programmbibliothek kann demnach nicht als installationsfähiges Produkt angesehen werden. Als Anwendungsbeispiel für ein vom Nutzer direkt installierbares Programm kann aber die OpenOffice-Suite angeführt werden, die in lauffähiger Version zum Download unter der Website http://www.openoffice.org angeboten wird. Das Risiko ist in diesen Fällen allerdings beschränkt, da die durch das Produkthaftungsrecht geschützten Schäden (Körperschaden) durch diese Software wohl kaum verursacht werden können. Theoretisch besteht ein Restrisiko nur hinsichtlich Datenverlusten von der Festplatte des privaten Nutzers. Ob Datenverlust als Sachschaden im Sinne des Produkthaftungsrechts qualifiziert werden kann, ist jedoch noch nicht restlos geklärt.

Ungenügend enger Kausalzusammenhang Wenn ein anderer als der ursprüngliche Entwickler die auf der öffentlichen Plattform angebotene Open-Source-Software zur Marktreife gebracht hat, ist die Kausalkette vom Fehler bis zur Haftung in aller Regel so lang, dass der ursprüngliche Beitrag des Entwicklers hinsichtlich der Fehlerverursachung in den Hintergrund tritt.

Software nicht im Rahmen der beruflichen Tätigkeit hergestellt Je nach Fallkonstellation kann der Entwickler mit dem Argument durchdringen, er habe nicht „für den Verkauf oder eine andere Form des Vertriebs mit wirtschaftlichem Zweck hergestellt" (Art. 5 Abs. 1 lit. c PrHG; vgl. auch § 1 Abs. 2 Ziff. 3 ProdHaftG; siehe Spindler (2004, S. 202) sowie Jaeger und Metzger (2006, Rz. 229)). Dabei sollte bejaht werden, dass beim Angebot zum kostenlosen Download in der Regel kein Vertrieb mit wirtschaftlichem Zweck vorliegt (vgl. Jaeger und Metzger (2006, Rz. 228) und Spindler (2004, S. 203)).

5.2 Bei Schädigung des Vertragspartners

Ist der direkte Vertragspartner geschädigt, so ist ergänzend das Argument entlastend, dass der Vertragspartner in die Software keine berechtigten Sicherheitserwartungen haben durfte, denn ein Fehler kann von vornherein nur dann vorliegen, wenn berechtigte Sicherheitserwartungen enttäuscht wurden (Taeger 1995, S. 102). Maßgeblich ist, ob das Produkt zum Gebrauch tauglich ist, mit dem vernünftigerweise gerechnet werden darf (Art. 4 Abs. 1 PrHG; § 3 Abs. 1 ProdHaftG).

Fehlerhaftigkeit im Sinne des Produkthaftpflichtrechts ist ein unbestimmter Rechtsbegriff. Erst im Konfliktfall stellt ein Richter fest, ob die Software von einem wie auch immer zu erwartenden Sicherheitsstandard abweicht oder nicht. Der Begriff der Fehlerhaftigkeit ist produkthaftungsrechtlich autonom auszulegen (siehe Straub (2002, Rz. 32) und Spindler (2004, S. 203 f.)). Dabei wird auf die Erwartungshaltung der Allgemeinheit abgestellt (siehe Straub (2002, Rz. 33 f., 39 ff.) und Spindler (2004, S. 203 f.)). Es ist ein objektiver Maßstab anzulegen (vgl. Fellmann 1996, N 3 zu Art. 4 PrHG). Entscheidend ist, welches Maß an Sicherheit eine verständige Person in Kenntnis und bei Abwägung aller relevanten Umstände für erforderlich hält. Von der Produkthaftung ausgeschlossen sind demnach Fehler, die nach dem „Stand der Wissenschaft und Technik" zum Zeitpunkt des Inverkehrbringens nicht erkannt werden konnten (Taeger 1996, S. 267). Bei Open-Source-Software kann dem Entwickler deshalb die Tatsache zugute kommen, dass die Entwicklung oft in einer großen Community diskutiert wird. *Community building* (Stürmer und Myrach 2006, S. 219 ff.) erhält hier eine zusätzliche, haftungsbeschränkende Komponente (Lemyre 2002, Rz. 105): Wenn eine ganze Vielzahl von Entwicklern den Fehler nicht bemerkt hat, so kann der Fehler nicht im Widerspruch zum Stand der Wissenschaft und Technik angesehen werden. Der Entwickler wird gleichwohl Vorsorge treffen müssen. Wenn er im Konfliktfall nachweisen kann, dass er die Software auf die kritischsten Fehler hin geprüft hat,

steht er deutlich besser da. Außerdem empfiehlt es sich zu dokumentieren, wie groß die Entwicklercommunity zum Zeitpunkt des Uploads der Open-Source-Software war und wie lebhaft die Diskussion fremder Codebestandteile zu diesem Zeitpunkt geführt wurde.

Dass vor der Marktreife berechtigte Sicherheitserwartungen in passiv vertriebene Open-Source-Software gesetzt werden könnten, ist auch deshalb ausgeschlossen, weil heute dem breiten Publikum bekannt sein dürfte, dass Open-Source-Software ständig weiterentwickelt wird und die Entwicklergemeinschaft deshalb keine Haftung und Gewährleistung für Funktionalität und Fehlerlosigkeit übernehmen will. Zusammenfassend kann basarartig entwickelte Open-Source-Software nicht gleichermaßen unter den Produktbegriff des Produkthaftpflichtrechts fallen wie proprietäre Software.

6 Produkthaftpflicht im aktiven Vertrieb

Im aktiven Vertrieb bestehen diese Argumente grundsätzlich nicht, auch wenn Open–Source-Software schadensverursachend war. Hier dürfte die Produkthaftpflicht grundsätzlich gleich zum Tragen kommen wie beim herkömmlichen proprietären Softwarevertrieb (Straub 2004). Wesentlich ist der Unterschied, dass der Anbieter der Open–Source-Software die Verwendungsart der Software kannte oder voraussehen konnte.

Bei genauer Betrachtung kann die Haftung aus PrHG/ProdHaftG allerdings auch nach aktivem Vertrieb nicht immer greifen. Open-Source-Software zeichnet sich dadurch aus, dass sie vom Empfänger weiterentwickelt werden darf. Das heißt, dass der Anbieter von Open-Source-Software im aktiven Vertrieb nicht gleichermaßen Kontrolle über das weitere Schicksal der Software hat wie ein Hersteller von proprietärer Software. Dieser Unterschied wirkt sich auf die produkthaftungsrechtliche Würdigung aus. Macht der Anwender von den aus einer Open-Source-Software–Lizenz abgeleiteten Freiheiten Gebrauch, müssen dem Anbieter wiederum die bereits erwähnten Einreden gegen eine Haftpflicht stehen offen (vergleiche Abschnitt 5). Mit anderen Worten lässt sich die Produkthaftung für Open-Source-Software auch nach aktivem Vertrieb nur dann bejahen, wenn der Anwender die Software gerade nicht wie Open-Source-Software benutzt. Wird aktiv vertriebene Open-Source-Software vom Empfänger weiterentwickelt, darf der Anbieter der ursprünglichen Version nicht verantwortlich gemacht werden für das Schädigungspotenzial, das von der weiterentwickelten Software ausgeht (a. M. aber Taeger (1996, S. 264 f.); kritisch hierzu Günther (2001, S. 210)). Im Haftungsprozess wird also relevant, ob der Anwender die Software unverändert oder erst nach einer dem Vertrieb folgenden Modifikation eingesetzt hat.

Somit kann ein Anbieter von Open-Source-Software unter Umständen der Produktehaftpflicht unterstehen. Dieses Risiko lässt sich vertraglich nicht wegbedingen, da die Produktehaftung immun ist gegen die Haftungsausschlussklauseln in Open–Source-Lizenzen. Der Geschädigte muss Ansprüche aus Produkthaftungsrecht innerhalb von drei Jahren nach Schadenseintritt (§ 12 ProdHaftG; Art. 9 PrHG), spätestens

jedoch innerhalb von zehn Jahren seit Inverkehrbringen der Software geltend machen (§ 13 ProdHaftG; Art. 10 PrHG).

Die praktischen Konsequenzen aus der Anwendbarkeit des Produkthaftungsrechts werden sich jedoch in Grenzen halten, da Software meistens „nur" Datenverluste verursacht, nicht aber Körperschäden. Datenverluste sind – wenn überhaupt – von vornherein nur dann ein nach dem Produkthaftungsrecht ersetzungsfähiger Schaden, wenn die Daten auf einem privat genutzten Computer gespeichert waren. Sollte Open-Source-Software aber ein erhöhtes Schädigungspotenzial aufweisen (insbesondere, wenn die Möglichkeit von Körperschädigungen nicht ausgeschlossen werden kann), wird der Anbieter sie nur dann freigeben, nachdem er jede einzelne Komponente auf Herz und Nieren geprüft hat – selbstverständlich mit der Folge, dass solche Anwendungen nur gegen einen erhöhten Preis zu haben sind.

7 Zusammenfassung

Das Risiko, dass sich für einen Entwickler von Open-Source-Software aus dem Produkthaftrecht ergibt, dürfte heute überbewertet werden. Die Rechtsordnung hält für Open-Source-Software grundsätzlich befriedigende Lösungen bereit. Der Rechtsanwender muss jedoch bereit sein, die Besonderheiten von Open-Source-Software zu berücksichtigen und auf Wertungsunterschiede einzugehen. Werden diese Unterschiede berücksichtigt, gelangt man zum Resultat, dass das Konzept der Produkthaftung auf Open-Source-Software nicht recht passt.

Im passiven Vertrieb kann es nur dann zur Produkthaftung kommen, wenn Open–Source-Software entweder als selbstständig laufende Anwendung (*executable*) verfügbar gemacht wird oder wenn eine Sondersituation vorliegt, insbesondere wenn der Anbieter die Software besonders anpreist und so erhöhte Sicherheitserwartungen weckt.

Auch bei aktiv vertriebener Open-Source-Software kann das Konzept der Produkthaftung nicht unbesehen angewendet werden. Produkthaftung scheidet insbesondere dann aus, wenn Open-Source-Software nach dem Vertrieb modifiziert wurde.

Literatur

Ann, C. (1991), *Die Produkthaftung des Lizenzgebers: Zur Haftungssituation von Innovationsverträgen im deutschen und im amerikanischen Recht*, Heymann, Köln.

Bartsch, M. (1998), *Software und das Jahr 2000 – Haftung und Versicherung für ein technisches Großproblem*, Nomos Verlagsgesellschaft, Baden-Baden.

Deike, T. (2003), 'Open Source Software – IPR-Fragen und Einordnung ins Deutsche Rechtssystem', *Computer und Recht* **19**(1), S. 9–18.

Fellmann, W. (1996), Kommentar zum Produktehaftpflichtgesetz, *in* H. Honsell, N. P. Vogt und W. Wiegand (Hrsg.), 'Obligationenrecht I, Basler Kommentar zum schweizerischen Privatrecht', 2. Aufl., Helbing & Lichtenhahn, Basel.

Günther, A. (2001), *Produkthaftung für Informationsgüter. Verlagserzeugnisse, Software und Multimedia im deutschen und US-amerikanischen Produkthaftungsrecht*, Verlag Otto Schmidt, Köln.

Hilty, R. M. (2001), *Lizenzvertragsrecht: Systematisierung und Typisierung aus schutz- und schuldrechtlicher Sicht*, Stämpfli, Bern.

Hoeren, T. (2004), Open Source und das Schenkungsrecht – eine durchdachte Liaison?, *in* R. Bork, T. Hoeren und P. Pohlmann (Hrsg.), 'Recht und Risiko', Bd. 2, Verlag Versicherungswirtschaft, Karlsruhe, S. 229–240.

Jaeger, T. und Metzger, A. (2006), *Open Source Software*, 2. Aufl., C. H. Beck, München.

Knop, C. (2003), 'Ein Juraprofessor schreckt die Linux-Welt auf', *FAZ* **13. Aug 2003**.

Koch, F. (2000), 'Urheber- und kartellrechtliche Aspekte der Nutzung von Open-Source-Software', *Computer und Recht* **16**(6), S. 333–344.

Lemyre, P.-P. (2002), Les logiciels libres sous l'angle de la responsabilité civile, Diplomarbeit, Université de Montréal. http://www.juriscom.net/uni/visu.php?ID=106 [19. Dez 2006].

Lenhard, F. (2006), *Vertragstypologie von Softwareüberlassungsverträgen*, Herbert Utz Verlag, München.

Lions, J. L. (1996), 'Ariane 5 – Flight 501 Failure', Bericht des Untersuchungsausschusses vom 19. Juli 1996. http://homepages.inf.ed.ac.uk/perdita/Book/ariane5rep.html [19. Dez 2006].

Lutterbeck, B., Horns, A. H. und Gehring, R. A. (2000), Sicherheit in der Informationstechnologie und Patentschutz für Softwareprodukt – ein Widerspruch?, Kurzgutachten für den Bundeswirtschaftsminister, TU Berlin.

Madison, M. (2003), 'Reconstructing the Software License', *Loyola University Chicago Law Journal* **35**(1), S. 275–340. http://www.law.pitt.edu/madison/papers/reconstructing_the_software_license.pdf [18. Dez 2006].

Morscher, L. (1999), 'Gewährleistung und Haftung bei der Software-Überlassung', *Schweizerische Zeitschrift für Wirtschaftsrecht* **1999**(2), S. 61–77.

Paul, A. G. (2005), Außervertragliche Haftung für Open Source Software, *in* 'GI Jahrestagung (1)', S. 131–136. http://trustsoft.uni-oldenburg.de:19080/30/01/ISOS-2005-Beitrag.pdf [18. Dez 2006].

Schiffner, T. (2003), *Open Source Software: freie Software im deutschen Urheber- und Vertragsrecht*, VVF-Verlag, München.

Sester, P. (2000), 'Open-Source-Software: Vertragsrecht, Haftungsrisiken und IPR-Fragen', *Computer und Recht* **16**(12), S. 797–807.

Spindler, G. (2003), Rechtsfragen der Open Source Software, Gutachten, Verband der Softwareindustrie Deutschlands e. V., München. http://lehrstuhl-spindler.uni-goettingen.de/extern/ross/downloads/studie_final.pdf [19. Dez 2006].

Spindler, G. (Hrsg.) (2004), *Rechtsfragen bei Open Source*, Verlag Otto Schmidt, Köln.

Straub, W. (2002), *Produktehaftung für Informationstechnologiefehler: EU-Produktehaftungsrichtlinie und schweizerisches Produktehaftungsgesetz*, Schulthess Verlag, Zürich.

Straub, W. (2004), *Informatikrecht*, Stämpfli Verlag, Bern.

Stürmer, M. und Myrach (2006), Open Source Community Building, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 219–234. http://www.opensourcejahrbuch.de/download/jb2006/chapter_05/osjb2006-05-02-stuermer-myrach [26. Dez 2006].

Taeger, J. (1995), *Außervertragliche Haftung für fehlerhafte Computerprogramme*, Mohr, Tübingen.

Taeger, J. (1996), 'Produkt- und Produzentenhaftung bei Schäden durch fehlerhafte Computerprogramme', *Computer und Recht* **12**(5), S. 257–271.

Wacha, J. (2005), 'Taking the Case: is the GPL enforceable?', *Santa Clara Computer & High Tech Law Journal* **21**, S. 451–492. [19. Dez 2006].

Weber, R. H. (2002*a*), Ali Baba oder das Risiko exklusiver Informationsinhaltsrechte, *in* R. J. Schweizer, H. Burkert und U. Gasser (Hrsg.), 'Festschrift für Jean Nicolas Druey', Schulthess, Zürich, S. 1009–1028.

Weber, R. H. (2002*b*), Freie Software – Befreiung vom Vertragstypenkonzept, *in* F. Harrer, W. Portmann und R. Zäch (Hrsg.), 'Besonderes Vertragsrecht – aktuelle Probleme: Festschrift für Heinrich Honsell zum 60. Geburtstag', Schulthess, Zürich, S. 41–59. http://www.rwi.unizh.ch/zik/news/Freie%20Software1.pdf [19. Dez 2006].

Weber, R. H. (2003*a*), Haftungsgrundlagen beim IT-Outsourcing, *in* R. H. Weber, M. Berger und R. Auf der Maur (Hrsg.), 'IT-Outsourcing', Schulthess, Zürich, S. 117–138.

Weber, R. H. (2003*b*), Open Source Software – Vertragsgestaltung, *in* R. H. Weber, M. Berger und R. Auf der Maur (Hrsg.), 'Geschäftsplattform Internet IV', Schulthess, Zürich, S. 73–92.

Widmer, M. J. (2003), *Open Source Software – Urheberrechtliche Aspekte freier Software*, Stämpfli, Bern.

Zirlick, B. (2003), *Freizeichnung von der Deliktshaftung*, Stämpfli, Bern.

Die GPL v3 – ein Schutzschild gegen das Damoklesschwert der Softwarepatente?

LINA BÖCKER

(CC-Lizenz siehe Seite 563)

Große Reformen hinterlassen Spuren. So auch die fast vollständige Neufassung der *General Public License (GPL)*.[1] Insbesondere patentrechtliche Fragen, eines der meist umstrittensten Gebiete im Bereich der Rechtsinformatik, wurden erstmals ausdrücklich geregelt und in den Lizenztext in Form der neuen Ziff. 11 aufgenommen. Ziel dieser Änderungen war es, bekannten Risiken, die aus dem Aufeinandertreffen von freier Software und Patentrecht entstehen, vorzubeugen und mehr Rechtssicherheit zu schaffen. Ziel dieses Beitrags ist es, diese Gefahren aufzuzeigen und zu überprüfen, ob die neuen Regelungen tatsächlich Abhilfe schaffen oder ob sie nur ein Placebo sind. Letzten Endes ist aber festzuhalten, dass auch durch die Neufassung der GPL nur die Symptome gelindert werden können. Wirklich helfen kann nur ein neu überdachtes, wenigstens europaweit harmonisiertes Patentrecht.

Schlüsselwörter: GPL · Lizenzrecht · Patentrecht · Softwarepatente

1 Bleibt alles anders – die neue GPL

Verständlicher und juristisch einfacher sollte sie werden, die neue GPL. „Die größte Gemeinschaftsarbeit, die jemals an einer Softwarelizenz vorgenommen wurde" (Jaeger 2006), hat zu einer fast vollständigen Neufassung der Bedingungen geführt, die das Ergebnis angeregter Diskussion aller Beteiligten ist. Besonders kontrovers wurden Kopier- (*Digital Rights Management*, *DRM*) und Patentschutz für Computerprogramme debattiert. Kritik an den auf Patente bezogenen Klauseln kam vor allem aus den Reihen der Softwareindustrie und hat, was Softwarepatente angeht, zu erhebli-

1 Dieser Artikel berücksichtigt den Diskussionsstand zur GPL v3 bis zum 27. Februar 2007.

chen Änderungen im Vergleich zur ersten Entwurfsfassung vom Januar 2006 geführt.[2] Während sich die GPL v2 in ihrer Präambel und auch noch der erste Diskussionsentwurf der GPL v3 vehement gegen Softwarepatente in jeglicher Form ausgesprochen haben, kritisiert die neue Fassung nur noch auf Universalrechner bezogene Patente.[3]

Der erste Diskussionsentwurf enthielt, anders als die „alte" GPL v2, die explizite Erteilung eines einfachen Nutzungsrechts am Patent an jeden Erwerber der Software, also eine Art automatische Patentlizenz (FSFE 2006). Diese Konstruktion wurde durch den verpflichtenden Verzicht auf die Geltendmachung von Patentansprüchen ersetzt. Einen übermäßigen Eingriff in die Rechte des Patentinhabers, die schließlich Eigentumsrechte sind, wollten die Verfasser verhindern.

Das Thema „Patente für computerimplementierte Erfindungen" ist in Industrie und Wissenschaft schon fast ein „Klassiker", wird doch seit mehr als 20 Jahren darüber diskutiert, ob und unter welchen Voraussetzungen Software bzw. Computerprogramme patentierbar sein sollen. Wer genau hinschaut, stellt fest, dass es einen ähnlichen Streit schon gab: Nach der Gründung des Zweiten Deutschen Reichs 1871 diskutierte man darüber, das Patentwesen insgesamt abzuschaffen, da es kein einheitliches gab und eine Vereinheitlichung schwierig erschien. In diesem Kontext kam auch die Diskussion auf, wie mit der Patentierbarkeit für Stoffzusammensetzungen auf dem Sektor der chemischen Industrie umgegangen werden sollte (Nack 2004, S. 774). Hier wurde mit Zustimmung der Industrievertreter ein Patentierungsverbot beschlossen, das trotz nachträglich auftretender Schwierigkeiten 90 Jahre bestehen blieb.

2 Die Diskussion über die Softwarepatente in der EU

Die Diskussion über das Ob und Wie eines Patentschutzes für Computerprogramme ist in den vergangenen zwei Jahren von einer rechtsdogmatischen zu einer rechtspolitischen geworden.[4] Für Außenstehende ist die Debatte insbesondere um den Entwurf einer europäischen *Richtlinie über die Patentierbarkeit computerimplementierter Erfindungen*[5] immer schwerer zu verstehen, was unter anderem damit zusammenhängt, dass viele der Streitigkeiten lediglich auf terminologische Ungenauigkeiten zurückzuführen sind. Zwar wurde der Richtlinienentwurf im Juli 2005 gekippt, das Thema ist jedoch weiterhin auf dem Tisch. Dass noch immer eifrig debattiert wird, liegt daran, dass der Zustand ohne Richtlinie ebenso wenig den Vorstellungen der „Kriegsparteien" entspricht wie der Zustand mit, da ohne ein Gesamtkonzept die Patentierungsvoraussetzungen insgesamt unklar und uneinheitlich sind. Das birgt natürlich auch die Gefahr einer Ausweitung des Patentschutzes über das notwendige Maß hinaus.

2 Der Begriff „Softwarepatente" wird im Rahmen dieses Artikels synonym für die „Patentierbarkeit computerimplementierter Erfindungen" verwendet.

3 Siehe dazu in der GPL-Präambel, letzter Absatz oder bei FSFE (2006).

4 Ausführlich dazu bei Horns (2000, Abs. 1–80).

5 Erster Entwurf im Amtsblatt der EU Nr. C 151 E vom 25. Juni 2002, S. 129, eine gründliche Diskussion der kontroversen Entwürfe findet sich bei Metzger (2003).

Die Frage nach der Möglichkeit der Patentierbarkeit von Computerprogrammen stellt sich in der oft behaupteten Form eigentlich gar nicht.[6] Denn bei der Frage nach der grundsätzlichen Patentierungsmöglichkeit kann es nicht um das äußere Erscheinungsbild einer Erfindung gehen (Nack 2004).[7] Zu prüfen ist vielmehr, welche erfinderische Leistung hinter dem in Frage stehenden Produkt steckt. Das gilt selbstverständlich auch für Computerprogramme. Grenzt das Patentgesetz in § 1 Abs. 3 Nr. 3 Computerprogramme als solche vom Schutz aus, so ist nichts anderes als das gemeint. Allein die Tatsache, dass es sich bei der Erfindung um ein Computerprogramm handelt, spricht weder für noch gegen ihre Patentierbarkeit (Nack 2004). Die erfinderische Leistung, die Grundlage des Patents ist, muss auf einem Gebiet der Softwaretechnologie liegen, dass als „Technik" anerkannt ist (Nack 2004, S. 772). Gemeint sind Steuerungs- und Regelungstechniken, die den Ablauf technischer Prozesse beeinflussen, wie etwa die Elektronik in einem Auto.[8]

3 Patentrechtsverletzungen und Open-Source-Software

Die Abgrenzungsschwierigkeiten zwischen patentierbaren Programmen und jenen, die lediglich Computerprogramme als solche sind, führen zu unkalkulierbaren Risiken, was zwischen Patentgegnern und -befürwortern weitgehend unumstritten ist. Diese Risiken, die im Folgenden dargestellt werden, können sich bei Open-Source-Software um ein Vielfaches heftiger auswirken als bei proprietärer Software, da bei Letzterer der nicht zugängliche Quellcode gründliche Patentrecherchen verhindert.[9]

3.1 Grundsätzliche Voraussetzungen einer Patentverletzung

Gefahren für Entwickler und Anwender entstehen durch das Risiko einer Patentverletzung beim Umgang mit freier Software, die Schadensersatz- und Unterlassungsansprüche nach sich ziehen kann.[10]

Verletzungstatbestände

Handelt es sich bei dem in Frage stehenden Softwarepatent um ein *Verfahrenspatent* im Sinne von § 9 Nr. 2 PatG, ist also ein bestimmter technischer Ablauf geschützt, so genügt jegliche Anwendung des patentierten Verfahrens, um eine Patentverletzung zu begehen. Angewendet wird ein Verfahren dann, wenn es bestimmungsgemäß gebraucht wird, d. h. die wesentlichen Schritte, die zum verfahrensgemäßen Erfolg

6 Siehe dazu Nitschke (2002). Zur grundsätzlichen Patentfähigkeit so genannter softwarebezogener Erfindungen siehe Metzger (2000). Zum grundsätzlichen Streit um den Schutz computerbezogener Erfindungen eingehend bei Weyand und Haase (2004).

7 Zur Eingrenzung des Begriffs *Softwarepatente* siehe Horns (2000).

8 Zum entscheidenden Merkmal der Technizität s. Horns (2000, Abs. 2 ff.) und Klopmeier (2002, S. 65ff.).

9 Siehe auch Horns (2000) sowie Weyand und Haase (2004).

10 Rechtsprechungsbeispiele bei Pierson (2004).

führen, müssen verwirklicht werden. Dies ist nur dann nicht verboten, wenn der Anwender berechtigter Benutzer des Verfahrens ist oder sich die Wirkung des Patents nicht auf seine Handlungen erstreckt. Bei Software reicht also das Ausführen auf einem Computer, um eine Patentverletzung herbeizuführen (Busse 2003, § 9 RdNr. 83 ff.).

Lange Zeit beschränkte sich der Softwareschutz auf solche Verfahrenspatente, da Computerprogramme in der Regel reale Prozesse nachbilden. Die jüngere Rechtsprechung lässt jedoch auch einen *Erzeugnisschutz* zu. Der Patentanspruch richtet sich dann nach dem digitalen Medium, auf dem das Programm gespeichert ist. Dies ist insoweit patentierbar, als es das Mittel zur Ausführung des technischen Verfahrens ist. Bereits das Herstellen, Anbieten oder Inverkehrbringen des geschützten Erzeugnisses ist hier rechtsverletzend,[11] also auch ein Anbieten über das Internet. Ebenfalls zur Benutzung eines Erzeugnispatents gehören die Einfuhr und der Besitz.

Nicht nur die direkte Verwendung des patentierten Erzeugnisses oder Verfahrens ist eine Patentverletzung. Eine mittelbare Benutzung im Sinne des § 10 PatG liegt bereits vor, wenn Mittel angeboten oder geliefert werden, die die unberechtigte Benutzung der Erfindung ermöglichen (Busse 2003, § 10 RdNr. 13).

Ausnahmetatbestände

Jeder dieser beschriebenen „Tatbestände" stellt eine Patentverletzung dar und führt zu Schadensersatz- bzw. Unterlassungsansprüchen. Nicht jedoch dann, wenn einer der patentrechtlichen Ausnahmetatbestände erfüllt ist, die sich aus § 11 PatG ergeben. Im Umgang mit Open-Source-Software kommt hier vor allem ein Handeln zu Versuchszwecken im Sinne von Nr. 2 in Frage. Das rechtfertigt die Verwendung patentierter Erzeugnisse dann, wenn es sich um die Verwendung von Code zur Neuentwicklung eines anderen Produkts handelt. Die Verwendung des Patents zu Versuchszwecken ist allerdings nur dann freigestellt, wenn es um die Gewinnung von mehr Erkenntnissen über die patentierte Erfindung geht.[12] Das ist dann nicht der Fall, wenn die Erfindung, also der patentierte Code, zum Mittel der Forschung gemacht wird. Die Weiterentwicklungsmöglichkeit darf Gegenstand dieser Patentverwendung sein, so dass die Open-Source-Weiterentwicklung grundsätzlich unter diesen Tatbestand fallen kann, auch wenn sie proprietären Code als Basis nimmt. Zu beachten ist dann aber, dass auch das Ergebnis der Forschung keine Patentverletzung darstellen darf.

Rechtsfolgen

Patentrechtsverletzungen haben Unterlassungsansprüche und im Verschuldensfalle – hier dem zivilrechtlichen Verschulden, welches auch die Fahrlässigkeit umfasst – auch Schadensersatzansprüche zur Folge. Es besteht einerseits die Gefahr einer Hemmung

11 Beispielsweise BGH GRUR 1959, 125 – Textilgarn; GRUR 1956, 77 – Rödeldraht; BGH GRUR 1979, 149, 151 – Schießbolzen; GRUR 1996, 109 – klinische Versuche.

12 Näher dazu Busse (2003, § 11 RdNr. 17 f.) mwN.; sowie GRUR 1996, 439, 446 und GRUR 1996, 120.

der Innovationsgeschwindigkeit bei der Geltendmachung von Unterlassungsansprüchen, andererseits birgt das Risiko einer Patentverletzung eine erhebliche finanzielle Bedrohung insbesondere für kleinere Entwickler, die auf eigene Faust arbeiten.

3.2 Mögliche Risiken der Patentierbarkeit von Software für Open Source

Was aber macht gerade den Patentschutz so gefährlich für die Open-Source-Entwicklung, dass er eines der Hauptthemen in der Diskussion um die GPL-Reform darstellt?[13] Zunächst bestehen gegen den Patentschutz von Software einige grundsätzliche Bedenken, unabhängig davon, ob es sich um Open Source oder um proprietäre Software handelt. Ein erteiltes Patent ist ein staatlich garantiertes Monopol, das es ermöglicht, eine Technologie unter Ausschluss jedes Dritten zu nutzen (Hufnagel 2002). Dieses Monopol besteht für 20 Jahre nach der Anmeldung der Erfindung beim Patentamt. Betrachtet man demgegenüber das Tempo, mit dem Entwicklungen in der Computerindustrie voranschreiten, erscheint diese Schutzdauer unverhältnismäßig lang. Schließlich sind ganze Programme oftmals bereits nach einem halben Jahr veraltet, ganz zu schweigen von einzelnen Programmteilen oder Updates und Patches. Die Konsequenz ist, dass die geschützten Teile „blockiert" sind, also für die Dauer des Schutzes nur mit Zustimmung des Patentinhabers verwendet werden dürfen.

Grundsätzlich ist der Patentschutz wie alle *Immaterialgüterrechte* zwar territorial begrenzt, d. h. auf das Gebiet desjenigen Landes beschränkt, in dem das Patent erteilt wurde. Durch internationale Abkommen sind allerdings die Patentsysteme der meisten Industriestaaten miteinander verbunden, so dass diese Territorialität relativiert wird. Es ist also relativ leicht, in mehreren Ländern Parallelpatente anzumelden (Hufnagel 2002, S. 280). Eine vollständige Abdeckung aller Staaten, in denen das Internet zugänglich ist, wird man dennoch nicht erreichen. Hinzu kommt, dass ein beträchtlicher Teil der Software über das Netz vertrieben wird. So entsteht die Gefahr unbewusster Patentverletzungen in Staaten, an die der Entwickler u. U. nicht gedacht hat.

Auch inhaltlich ist der Patentschutz vergleichsweise weit. Während das Urheberrecht nur die konkrete sprachliche Ausdrucksform eines Programms, nämlich den Quellcode selbst schützt, betrifft das Patentrecht die im Programm enthaltene technische Lehre in ihrer Gesamtheit, inklusive gleichwirkender Abweichungen im Quelltext. Geschützt ist hier die Funktionalität des Programms, nicht der konkrete Ausdruck dieser Funktionalität. Die Weite des Patentschutzes führt zu verschiedenen Risiken, die die Open-Source-Entwicklung tatsächlich hemmen können (Wiebe 2004*b*).

Lizenzpflicht für die Entwicklung

Besteht die grundsätzliche Möglichkeit, Software auch bei geringer Erfindungshöhe patentieren zu lassen, so kann dies dazu führen, dass geschäftstüchtige Entwickler (Unternehmen) sich schon kleinste Entwicklungsschritte schützen lassen. Auf diese

13 Eine eingehende Analyse des Konfliktpotenzials findet sich bei Wiebe (2004*a*) und Wiebe (2004*b*).

Weise entsteht eine Blockade, die nur von großen Unternehmen mit entsprechendem finanziellen Hintergrund überwunden werden kann.

Probleme bei der Recherche

Hinzu kommt, dass gerade für kleinere Entwickler und Unternehmen eine umfangreiche Patentrecherche schwierig und teuer werden kann, so dass schon das Einholen der Lizenzen ein unüberwindliches Hindernis darstellen kann. Je kleiner das Patent, desto umfangreicher und auch teurer ist die Recherche. Umso schwieriger wird die Nachforschung, weil der Quellcode nach derzeitiger Rechtslage nicht zur *Erfindungsbeschreibung*, also dem Text, der den Umfang des Schutzes definiert, gehört. Die konkreten Anweisungen des Programms sind also nicht erkennbar, so dass alles unter den Schutz fällt, das den gleichen Ablauf hervorruft. Eine „wasserdichte" Patentrecherche ist demnach kaum möglich, wenn der Entwickler nicht bei einem größeren Unternehmen angestellt ist. Daneben dürfte der hohe Zeitaufwand ein Entwicklungshindernis darstellen.

Probleme bei der unternehmerischen Patentrechtspolitik von Open-Source-Unternehmen

Angesichts der zunehmenden Patentierung von Softwarebausteinen insbesondere in den USA sind Praktiken wie die Errichtung von Patentpools und das so genannte *cross-licensing*, bei denen sich Unternehmen ihre Patente wechselseitig lizenzieren, weit verbreitet und finden zunehmend mehr Anerkennung. Eine Teilnahme an diesen Mechanismen ist aber nur möglich, wenn eigene Patente mit eingebracht werden können. Open-Source-Entwickler melden, meist wegen der grundsätzlichen Ablehnung von Patenten in der Präambel der GPL, in der Regel keine Patente an. Da sie deshalb auch keine eigenen Rechte einbringen können, können sie keinen Nutzen aus diesen Formen der Zusammenarbeit ziehen. Im Gegenteil werden sie von der Nutzung zahlreicher zum Teil sehr allgemeingültiger Softwarebausteine ausgeschlossen, die in solchen Pools enthalten oder im Wege des *cross-licensing* freigegeben werden. Dadurch wird nicht nur das Risiko einer Patentverletzung größer, sondern auch der freie Ablauf der Entwicklung gehemmt.

Patentverletzung durch Open-Source-Entwickler

Die systembedingte Offenheit des Quellcodes kann sich schon grundsätzlich als großer Nachteil im Hinblick auf Verletzungen geistigen Eigentums an Software erweisen.[14] Denn während bei Open-Source-Programmen Patent- sowie Urheberrechtsverletzungen einfach erkannt werden können, da über den Quellcode leicht Einblick in die Strukturen des Programms genommen werden kann, ist eine derartige „Patentrecherche" bei nur im Binärcode vertriebenen Programmen nicht möglich.

14 Vgl. dazu ausführlich Lutterbeck und Gehring (2004) und Metzger (2000).

Aufdeckung und Nachweis einer Rechtsverletzung sind bei Open Source deshalb sehr einfach. Das gilt nicht nur für vorsätzliche Verletzungen, sondern auch für unbeabsichtigte und sogar unbewusste. Proprietäre Software ist also doppelt im Vorteil: Weil der Quellcode geheim gehalten wird und eine Dekompilierung zur Aufdeckung einer Patentverletzung nach §§ 69c iVm § 69e Abs. 1 UrhG verboten ist, ist der Nachweis einer Patentverletzung nur schwer oder gar nicht möglich. Andersherum können aber die Inhaber von Patenten an proprietärer Software einfach feststellen, ob ein freies Programm geistiges Eigentum verletzt, da sie Einblick in den Quellcode haben, den sie selbst Dritten verwehren. Für einen Großteil der Open-Source-Entwickler kann diese Tatsache existenzbedrohend sein, da sie häufig unentgeltlich und in ihrer Freizeit arbeiten, sich aber kostenpflichtigen Abmahnungen und Prozessen ausgesetzt sehen, denen sie nichts entgegensetzen können (Nitschke 2002).

Patentanmeldung durch kommerzielle Unternehmen

Patentanmeldungen an im Wesentlichen aus OS-Bestandteilen bestehender Software durch kommerziell ausgerichtete Unternehmen können ebenfalls zu Konflikten führen. Einerseits können auf diese Weise „versteckte" Lizenzen an einem Programm entstehen, die deshalb gefährlich sind, weil sie zu Ansprüchen gegen alle Mitglieder der Vertriebskette führen können. Zudem erlangen die Inhaber der Patente die Macht über den Fortgang des Entwicklungsprozesses, weil sie durch ihre Lizenzpolitik die Verwendung einzelner Bestandteile verhindern können.

Patentanmeldung durch Open-Source-Entwickler

Aber auch aus den „eigenen Reihen" droht Gefahr. Die GPL v2 spricht sich zwar in der Präambel gegen Softwarepatente aus, hindert einen Entwickler aber dennoch nicht daran, ein Patent auf seine Software anzumelden und anschließend andere Nutzer patentrechtlich in Anspruch zu nehmen. Im Rahmen der GPL v2 wird deshalb angenommen, dass die Lizenz eine implizite, also nicht ausdrückliche, *Nutzungsrechtseinräumung* an dem Patent mit umfasst. Das ist deshalb problematisch, weil eine solche konkludente Lizenzerteilung nicht in allen europäischen Mitgliedstaaten anerkannt ist[15] und weil hierdurch u. U. in rechtswidriger Weise in das Recht des Patentinhabers eingegriffen wird, über den Umfang der Lizenzierung selbst zu bestimmen.

4 Abhilfeversuche in der GPL v3

Die neue Lizenz will diesen Schwierigkeiten so weit wie möglich begegnen und den Umgang mit Patenten für Softwarenutzer bzw. -entwickler erleichtern (Jaeger 2006). Im Gegensatz zur GPL v2, die nach allgemeiner Auffassung nur eine implizite Mitlizenzierung der dem Urheber zustehenden Patente enthielt (siehe FSF 2006, S. 3) und

15 Dieses Problem haben auch die Entwickler erkannt, vgl. FSF (2006, S. 17).

sich ansonsten über den Umgang mit möglicherweise am Programm bestehenden Patenten ausschwieg, beinhaltet die GPL v3 mehrere Regelungen, die das Verhältnis der freien Programme zu Softwarepatenten betreffen: Die aktuelle Fassung enthält vor allem einen Verzicht auf die Geltendmachung von Patentansprüchen durch den Weitergebenden, soweit sie diesem zustehen.

Darüber hinaus muss jeder, der GPL-Software weitergeben will und weiß, dass an ihr Patentrechte Dritter bestehen, seine Abnehmer entweder vor der Inanspruchnahme durch den Patentinhaber schützen oder sicherstellen, dass bei der Veröffentlichung des Codes die relevanten Informationen mitgeliefert werden. Dadurch ist er gezwungen, die notwendigen Informationen bezüglich der bestehenden Patente zu veröffentlichen. Ob es stets möglich ist, den Quellcode auf einem öffentlichen Server bereitzustellen, wenn an ihm Schutzrechte bestehen, ist in Zweifel zu ziehen. Wie der Schutz Dritter rechtlich konstruiert werden kann, so dass den Interessen aller Beteiligten wirksam gedient ist, wird ebenfalls Gegenstand der juristischen Diskussion sein müssen.

Es ist vorgesehen, dass jeder Empfänger von GPL-Software automatisch ein einfaches patentrechtliches Nutzungsrecht eingeräumt bekommt. Die erste GPL-Fassung enthielt noch ein solches Nutzungsrecht in ausdrücklicher Form. Da man aber nicht zu stark in die Rechte des Rechteinhabers eingreifen wollte, hat man die ausdrückliche Klausel wieder entfernt. Von einer impliziten Lizenz ist aber weiterhin auszugehen.[16] Die neue Ziffer 2 enthält zudem die Auflage, dass der Nutzer keine *Patentansprüche* bezüglich des betreffenden Programms geltend macht. Anderenfalls verliert er seine Rechte aus der Lizenz, so, wie das in der GPL v2 schon bislang der Fall war, wenn gegen ihre Bedingungen verstoßen wurde. Auf diese Weise soll unter anderem verhindert werden, dass Entwickler eigene Fortentwicklungen „heimlich" patentieren lassen und dann andere Nutzer oder Entwickler, die gleichwertige Änderungen vornehmen, rechtlich verfolgen. Der *Rechterückfall* war in Bezug auf die Verletzung der urheberrechtlichen Nutzungsbefugnisse schon in der zweiten Version umstritten.[17]

Daneben lässt die GPL v3 eine Kombination mit Lizenzen zu, die noch weitergehende *Patentkonterklauseln* enthalten. Bemerkenswert ist hier, dass sich die Pauschalkritik an Softwarepatenten laut Präambel nun ausschließlich auf Patentierungen von Software für *general-purpose computers* bezieht. Hardwareimplementierte Software wie beispielsweise die Firmware eines Kernspintomographen, die eine fest verbundene Einheit mit der Hardware bildet, soll auch nach Auffassung der GPL-Entwickler patentierbar sein.

5 Wirksames Heilmittel oder nur Symptomtherapie?

Sind aber nun die Änderungen eine wirksame Arznei gegen die durch die Patentierung von Computerprogrammen entstehenden Gefahren oder bekämpfen sie nur

16 Vgl. dazu mit http://www.fsf.org/news/gplv3-clarification.

17 Dazu statt vieler: Schiffner (2002), Spindler (2004) und Jaeger und Metzger (2006).

die Symptome, ohne auf das eigentliche Problem einzugehen? Da ist zum einen der Verzicht auf die Geltendmachung von Patentansprüchen durch denjenigen, der die Software weiterverbreitet. Hier stellt sich eine praktische Schwierigkeit, die aus juristischen Feinheiten resultiert. Patente bleiben nämlich nicht immer in der Hand des Erfinders, sondern können auch wie eine Sache verkauft werden. Diese Tatsache kann und wird im Zusammenhang mit dem Verzicht auf die Geltendmachung von Patentansprüchen zu Schwierigkeiten führen. Ein solcher Verzicht kann nämlich nur schuldrechtlich wirken, d. h. nur zwischen den Parteien, die ihn tatsächlich vereinbart haben, nicht gegenüber jedermann. Wird ein Patent also an jemand anderen verkauft, ist der Käufer dem Lizenznehmer gegenüber nicht mehr verpflichtet, auf die Geltendmachung von etwaigen Verletzungsansprüchen zu verzichten und kann diesen gegebenenfalls patentrechtlich verfolgen. Allein der Verzicht auf die Geltendmachung von Patentansprüchen reicht also nicht aus, um einen wirkungsvollen Schutz des Abnehmers zu gewährleisten.

Das kann schon eher durch die Verpflichtung zum Schutz der Abnehmer vor Patentansprüchen gelingen,[18] deren juristische Konstruktion allerdings wie bereits angesprochen ungeklärt ist. In Betracht kommt erst einmal eine *befreiende Schuldübernahme*, bei der sich der Weitergebende seinem Abnehmer gegenüber verpflichtet, eine bei diesem entstehende Schuld vollständig zu übernehmen. Diese Konstruktion hat den Nachteil, dass sie gemäß § 415 BGB vom Patentinhaber als Gläubiger genehmigt werden muss, dessen Feststellung gerade bei Open-Source-Software manchmal recht aufwändig sein kann. Einfacher wäre ein „Schuldbeitritt mit Freistellungsverpflichtung", bei dem der Weitergebende der Schuld des Patentverletzers beitritt, also mit haftet, und im Innenverhältnis, also nur im Verhältnis zwischen den beiden, verpflichtet ist, seinen Abnehmer freizustellen. Zwischen beiden entsteht eine *Gesamtschuld*, es können also beide vom Patentinhaber in Anspruch genommen werden. Der Empfänger der Software kann sich sein Geld von dem Weitergebenden zurückholen, umgekehrt aber nicht. Daneben kann man auch noch annehmen, Distributor und Empfänger schlössen einen „konkludenten Garantievertrag" ab. Dabei gibt der Weitergebende, ohne es ausdrücklich zu sagen, eine Garantie dafür ab, dass der Nutzer nicht für eventuelle Patentverletzungen haften muss. Wird er dann doch in Anspruch genommen, so ist der Weitergebende auf Grund des Garantievertrages verpflichtet, den Betrag zu übernehmen. Welche dieser Konstruktionen sich letztendlich durchsetzen kann, wird stark von den jeweiligen Einzelverhältnissen abhängen. Dieses Schutzinstrument scheint zumindest in Bezug auf solche Patentansprüche, von denen der Distributor weiß, recht wirkungsvoll. Es kann aber nicht davon ausgegangen werden, dass auf diese Weise vollständige Sicherheit des Empfängers vor der Inanspruchnahme von Patentinhabern gewährleistet werden kann. Denn einerseits muss damit gerechnet

18 Die praktische Bedeutung dieser Verpflichtung muss sich allerdings erst noch zeigen. Die Möglichkeit, die notwendigen Informationen auf den Server zu laden, auf dem die Software angeboten wird, stellt einen erheblich geringeren Aufwand dar und wird deshalb vielleicht mehr Erfolg haben.

werden, dass es Patente gibt, von denen der Distributor nichts weiß und andererseits ist er nicht gezwungen, sich auf diese Auflage einzulassen. Er kann stattdessen auch den Code auf einem öffentlichen Server zur Verfügung stellen und damit die notwendigen Informationen über gegebenenfalls bestehende Patente veröffentlichen. Das wiederum wird ohne Patentverletzung nicht immer möglich sein, so dass auch hier kein vollständig wirksamer Schutz des Abnehmers gewährleistet werden kann.

Die Möglichkeit einer konkludenten Lizenzerteilung betrifft wiederum nur solche Nutzungsrechte, hinsichtlich derer der Distributor verfügungsbefugt ist. Er kann keine Lizenz erteilen für solche Patente, die jemand anderem gehören. Juristisch stellt es sich auch als problematisch dar, ob man eine konkludente Lizenz, die sich allenfalls aus einer Auslegung der konkreten Vereinbarung ergeben kann, neben einer ausdrücklichen Regelung wie der neuen Ziff. 11 tatsächlich wird annehmen können. Hinzu kommt, dass die Wirksamkeit einer solchen impliziten Lizenz im europäischen Rechtsraum zum Teil fraglich ist. Die Möglichkeit einer konkludenten Lizenzerteilung schadet also nicht, hilft aber auch nicht unbedingt gegen die Risiken von Softwarepatenten.

Am wirkungsvollsten, aber juristisch zugleich auch am schwierigsten zu beurteilen ist die Regelung, nach welcher der Nutzer alle Rechte aus der GPL verliert, wenn er *Patentansprüche* bezüglich der lizenzierten Software geltend macht. Mit dieser Klausel soll die heimliche Patentierung vermieden werden. Sie erinnert stark an die GPL v2, in der sich der Rechtewegfall auf eine Verletzung der urheberrechtlichen Bedingungen der Lizenz stützte. Schon hier waren die rechtliche Konstruktion und die Wirksamkeit eines solchen, „dinglichen", also gegenüber jedermann wirksamen, Rechtsverlustes ungeklärt. Letztendlich bestehen hier keine Unterschiede im Vergleich zu der ursprünglichen Regelung, die von der herrschenden Meinung als auflösend bedingte *Nutzungsrechtseinräumung* angesehen wurde. Auch die Geltendmachung von Patentansprüchen stellt eine auflösende Bedingung hinsichtlich der durch die GPL eingeräumten Rechte dar. Insofern kann auf die Diskussion zur GPL v2 verwiesen werden, wonach die Wirksamkeit dieser Konstruktion umstritten ist.[19] Endgültig sicherer Schutz vor heimlicher Patentierung durch Open-Source-Entwickler kann damit durch diese Regelung nicht gewährleistet sein.

Die erweiterte Kompatibilität der GPL, die nunmehr auch eine Kombination mit Software zulässt, die unter einer Lizenz mit mehr Möglichkeiten zu *Patentkontern* steht, hat letztlich auf die Sicherheit und den Schutz vor den genannten Risiken keinerlei Auswirkungen und verdeutlicht nur, dass die Verfasser der GPL Softwarepatenten nach wie vor kritisch gegenüberstehen.

6 Fazit

Tatsächlich birgt die Patentierung von Software einige Risiken für Open-Source–Entwickler und -Anwender. Zu nennen ist insbesondere die schwierige Recherche,

19 Nachweise siehe oben unter Abschnitt 4.

während gleichzeitig Patentverletzungen durch Open-Source-Entwickler durch den offen zugänglichen Quellcode leicht erkennbar sind. Gegen die besonders in den USA verwurzelte Praxis der Patentrechtspools kann die GPL nur insofern helfen, als sie die Patentierung auch von freier Software ausdrücklich zulässt, aber verlangt, dass jedem Nutzer eine unentgeltliche Lizenz an den betreffenden Patenten eingeräumt wird. Ob dadurch ein in Patentpools nützliches Zahlungsmittel entstehen kann, lässt sich allerdings auf einer rein theoretischen Ebene nur schwer beantworten.

Durch die Recherchehindernisse entsteht ein weiteres Problem, nämlich das der Patentverletzung durch Open-Source-Entwickler. Da es schwierig ist, herauszufinden, welche Codebestandteile schon patentiert sind, erhöht sich natürlich das Risiko, beim Entwickeln eine Patentverletzung zu begehen. Diesem Risiko können auch die Neuregelungen in der GPL v3 nicht abhelfen, da der Distributor nur insoweit Schutz gewähren kann und muss, als er von den bestehenden Patentansprüchen weiß. Ist dies nicht der Fall, so gibt es auch keinen Schutz.

Zumindest eine abschreckende Wirkung werden die Neuregelungen aber auf solche Entwickler freier Software haben, die sich selbst mit dem Gedanken tragen, ihre Entwicklungen patentieren zu lassen. Das ist zwar grundsätzlich möglich, die Gefahr, die daraus für andere Entwickler entsteht, wird jedoch durch die Regelungen in der neuen GPL gebremst. Wer solche Software weitergibt, muss darauf verzichten, die ihm zustehenden *Patentansprüche* gegenüber den Nutzern und Weiterentwicklern des Programms geltend zu machen, wenn er es unter der GPL lizenzieren will. Ebenso kann sich aus den Umständen eine konkludent erteilte einfache Lizenz ergeben. Hinzu kommt, dass jeder Distributor seine Abnehmer vor solchen Patentansprüchen schützen muss, von denen er weiß, also auch vor seinen eigenen. Tut er das nicht, so muss er die Informationen über bestehende Patente und den Code veröffentlichen, so dass auch hier keine Unsicherheit mehr bestehen kann.

Im Ergebnis lässt sich also festhalten, dass die Neuregelungen in der GPL v3 zwar wirkungsvoll vor der Gefahr aus den eigenen Reihen schützen, das eigentliche Problem, nämlich das Dilemma zwischen schwieriger Patentrecherche bei proprietärer Software und dem offenen Code können sie jedoch nicht vollständig lösen. Das wird letztendlich Aufgabe des Gesetzgebers sein, der ein Patentrecht schaffen muss, das kommerzielle und Open-Source-Interessen in Einklang bringt.[20]

Literatur

Busse, R. (2003), *Patentgesetz, Kommentar*, 6. Aufl., de Gruyter, Berlin.

FSF (2006), 'GPLv3 First Discussion Draft Rationale', Free Software Foundation. http://gplv3.fsf.org/gpl-rationale-2006-01-16.pdf [12. Jan 2007].

20 Zu den rechtspolitischen Optionen siehe Horns (2000). Vorschläge gibt es auch bei Weyand und Haase (2004, S. 203) und bei Lutterbeck und Gehring (2004, S. 12 ff.). Die ausführliche Diskussion dieser Vorschläge würde den Rahmen dieses Beitrags allerdings bei Weitem sprengen.

FSFE (2006), 'Patents and GPLv3', Free Software Foundation Europe. http://www.germany.fsfeurope.org/projects/gplv3/patents-and-gplv3.en.html [02. Jan 2007].

Horns, A. H. (2000), 'Der Patentschutz für softwarebezogene Erfindungen im Verhältnis zu „Open Source"-Software', JurPC. http://www.jurpc.de/aufsatz/20000223.htm.

Hufnagel, F. (2002), 'Software- und Business-Patente – Herausforderung für das juristische Risikomanagement', *Multimedia und Recht* (5), S. 279 ff.

Jaeger, T. (2006), 'Eine Lizenz entsteht – Neue Entwürfe für GPL und LGPL', heise open. http://www.heise.de/open/artikel/76248 [12. Dez 2006].

Jaeger, T. und Metzger, A. (2006), *Open Source Software. Rechtliche Rahmenbedingungen der Freien Software*, 2. Aufl., C. H. Beck, München.

Klopmeier, F. (2002), 'Zur Technizität von Software', *Mitteilungen der deutschen Patentanwälte* **93**(2), S. 65–70.

Lutterbeck, B. und Gehring, R. A. (2004), Software-Patente im Spiegel von Softwareentwicklung und Open Source Software, *in* J. Taeger und A. Wiebe (Hrsg.), 'Informatik – Wirtschaft – Recht: Regulierung in der Wissensgesellschaft, Festschrift für Wolgang Kilian', Nomos, Baden-Baden, S. 301–322. http://ig.cs.tu-berlin.de/ma/rg/ap/2003-x/GehringLutterbeck-SWPat-092003.pdf [02. Jan 2007].

Metzger, A. (2000), 'Softwarepatente als Gefahr', *Linux Magazin* (7), S. 64–65. http://www.ifross.de/ifross_html/art4.html [02. Jan 2007].

Metzger, A. (2003), 'Softwarepatente im künftigen europäischen Patentrecht', *Computer und Recht* **19**(5), S. 313–317. http://www.ifross.de/ifross_html/art45.pdf [02. Jan 2007].

Nack, R. (2004), 'Neue Gedanken zur Patentierbarkeit von computerimplementierten Erfindungen: Bedenken gegen Softwarepatente – ein déjà vu?', *GRUR Int* (9), S. 771–775.

Nitschke, T. (2002), 'Geistiges Eigentum ist Diebstahl. Freie Software vs. Software-Patente', *Forum Recht Online* (3), S. 91–93. http://www.forum-recht-online.de/2002/302/302nitschke.htm [02. Jan 2007].

Pierson, M. (2004), 'Softwarepatente – Meilensteine der patentrechtlichen Rechtsprechung', JurPC. http://www.jurpc.com/aufsatz/20040182.htm [13. Jan 2007].

Schiffner, T. (2002), *Open Source Software. Freie Software im deutschen Urheber- und Vertragsrecht*, 1. Aufl., VVF, München.

Spindler, G. (2004), *Rechtsfragen bei Open Source*, Otto Schmidt, Köln.

Weyand, J. und Haase, H. (2004), 'Anforderungen an einen Patentschutz für Computerprogramme', *GRUR Int* (3), S. 198 ff.

Wiebe, A. (2004*a*), 'Softwarepatente – das Ende von Open Source?', *Medien und Recht* (3), S. 195 ff.

Wiebe, A. (2004*b*), 'Softwarepatente und Open Source', *Computer und Recht* **20**(12), S. 881–888.

Von der Kommune zur Community: Freie Netze und freies Wissen auf der lokalpolitischen Agenda

LEONHARD DOBUSCH

(CC-Lizenz siehe Seite 563)

Während auf nationaler und supranationaler Ebene die Förderung oder Behinderung der diversen sozialen Bewegungen rund um Freie und Open–Source-Software schon seit langem Gegenstand von politischen Auseinandersetzungen sind, geht ihre lokale Fundierung und Relevanz in der Regel mit kommunalpolitischer Ignoranz einher. Unter anderem am Beispiel der Europäischen Kulturhauptstadt 2009 – Linz/Donau – werden die kommunalen Handlungsmöglichkeiten für Beiträge in exemplarischen Anwendungsbereichen wie freie Inhalte (Creative Commons, Open Access, OpenCourseWare), Bekämpfung des *digital divide* und Freie/Open-Source-Software skizziert.

Schlüsselwörter: Kommunalpolitik · freies Wissen · freie Inhalte · Linz 2009

1 Intro

„Vorbildwirkung beginnt auf lokaler Ebene." (Lawrence Lessig)

„Labor der Zukunft." – Unter diesem Motto bewarb sich Linz/Donau, die drittgrößte Stadt Österreichs, erfolgreich um den Titel „Europäische Kulturhauptstadt des Jahres 2009". Ob sie diesem Anspruch im Bereich digitaler Freiheiten gerecht werden kann, wird sich aber schon in den Jahren vor 2009 zeigen. Zwar sind Organisationen wie die *Free Software Foundation (FSF)* oder *Creative Commons (CC)* Preisträger der jährlich im Rahmen der *Prix Ars Electronica* in Linz vergebenen *Goldenen Nica*, auf Ebene profaner Kommunalpolitik hat sich deren Anerkennung aber bis dato auch in Linz nur sehr spärlich niedergeschlagen. Egal ob es um Freie/Open-Source-Software (FOSS), Zugang zu wissenschaftlichem oder didaktischem Wissen in Form von Open Access bzw. *OpenCourseWare* oder ganz allgemein um freie Inhalte geht – nirgends

nimmt die Stadt Linz oder die Linzer Universität eine herausragende Stellung ein. Trotzdem soll im Folgenden – neben anderen lokalen Initiativen – vor allem am Linzer Beispiel über die kommunalpolitischen Potenziale oder sogar Imperative reflektiert werden, die sich im Kontext der diversen sozialen Bewegungen für freie Netze und freies Wissen auf lokaler Ebene eröffnen.

Hauptgrund dafür ist eine umfangreiche Sammlung von Themen und Projekten unter der Schirmherrschaft der Intendanz des Linzer Kulturhauptstadtjahres 2009, in deren Rahmen 17 Menschen (unter Einschluss des Autors) an genau dieser Fragestellung arbeiten. Diese Sammlung ist Anfang dieses Jahres in Buchform erschienen.[1] Daneben bietet sich der Fall einer von Größe (ca. 200 000 Einwohner) und Ausgangsposition durchschnittlichen Kleinstadt auch deshalb an, weil er allgemeine Möglichkeiten und Grenzen kommunaler Einflussnahme auf scheinbar globale Vorgänge deutlich macht. Gerade in der Durchschnittlichkeit des Falls liegt seine Beispielhaftigkeit.

2 Leuchttürme digitaler Freiheit

Obwohl sich die verschiedenen Ausprägungen und Abkömmlinge der Bewegung für freie Software alle erst im Kontext der globalen Internetinfrastruktur gebildet haben, sind es mehr denn je lokale Initiativen und Pilotprojekte, die einen Unterschied machen. Der Slogan der Globalisierungsbewegung – „Global denken, lokal handeln" – gilt gerade im digitalen Bereich, wo mit geringen Kosten und binnen Sekunden ein lokaler Einzelfall zum weltweiten Vorbild erwachsen kann. Nirgends wird das deutlicher als im fast schon „klassischen" Fall von Software.

Im Markt für Desktop-Betriebssysteme wirken besondere ökonomische Mechanismen[2], die im Falle von proprietärer Software letztlich zu einem Anbietermonopol wie dem von *Microsoft Windows* geführt haben. Von Netzwerkeffekten spricht man immer dann, wenn der Nutzen des individuellen Anwenders mit der Gesamtzahl der Anwender steigt. Je mehr Menschen beispielsweise ein Betriebssystem einsetzen, desto mehr Firmen bieten dafür Zusatzprodukte und Spezialdienste an und desto mehr qualifizierte Arbeitskräfte sind in der Regel verfügbar. Viele fachspezifische Softwareprogramme, die nur einen relativ kleinen Markt bedienen, werden in der Folge nur für ein Betriebssystem – eben den Marktführer – entwickelt. Was den Einzelnen noch wenig betreffen mag, wird spätestens bei größeren Organisationen mit verschiedenen Abteilungen und jeweils unterschiedlichen Fachanwendungen zum Problem.

Die Entscheidung der Münchner Stadtverwaltung im Jahr 2003, ihre komplette Desktop-Softwarelandschaft auf freie und Open-Source-Software umzustellen (Hoegner 2006) erscheint vor diesem Hintergrund weit weniger trivial. Erst im Zuge der

1 Siehe auch Dobusch und Forsterleitner (2007), welches unter einer CC-BY-SA-Lizenz online unter http://www.freienetze.at verfügbar ist.

2 Hierbei handelt es sich um die sog. „Netzwerkeffekte", siehe auch Shapiro und Varian (1999) sowie den Artikel „Die Ökonomie der Softwaremärkte" auf S. 125 in diesem Buch.

Umsetzung wurde den Verantwortlichen die Komplexität des Vorhabens in seiner ganzen Tragweite bewusst: Neben hunderten Fachanwendungen und Tausenden Makros, die es umzustellen galt, war die gesamte IT-Struktur der Stadt komplementär zum verwendeten Microsoft-Betriebssystem. Zur technischen Umstellung gesellte sich die Notwendigkeit einer organisatorischen Restrukturierung. Aber gerade in der Überwindung dieser Schwierigkeiten beweist die Stadt München mit ihrem Migrationsprojekt der ganzen Welt, dass ein Umstieg technisch und wirtschaftlich machbar ist. Und die Besuchergruppen aus Spanien, Südkorea und vielen anderen Ländern zeigen, dass die Sichtbarkeit lokaler Initiative in einer globalisierten Welt enorm gestiegen ist. Auch die teilweise Linux-Migration in der Wiener Stadtverwaltung (Lutz 2006) wurde zumindest auf politischer Ebene wesentlich durch die Münchener Entscheidung beschleunigt.

So wie die Münchner Stadtverwaltung im Bereich der Desktop-Software inzwischen Nachfolgern als „Leuchtturm" dient, gibt es auch in anderen Bereichen bereits lokale Vorreiter.

3 Freie Netze

Weit wichtiger als die in einer Stadtverwaltung verwendete Softwareumgebung ist für die Bevölkerung mit Sicherheit der technische und soziale Zugang zu den neuen digitalen Möglichkeiten und Freiheiten. Was nützen all die freien Inhalte in der *Wikipedia*, was bringt all die im Netz verfügbare freie Software denjenigen, die keinen Zugang zum Internet haben? Inzwischen haben einige Städte erkannt, dass der Kampf gegen die „Digital Divide" genannte Ungleichheit im Zugang zu neuen Informationstechnologien (auch) auf kommunaler Ebene geführt werden muss.

3.1 Beispiele kommunaler Internet-Grundversorgung

Neben den vor allem im ländlichen Raum vorhandenen technischen Barrieren, sind es vor allem wirtschaftliche (insb. Kosten für PC und Internetzugang) und soziokulturelle (z. B. Bildung, Wahrnehmung neuer technischer Möglichkeiten) Faktoren, die gesellschaftliche Ungleichheiten auch in die digitale Welt des Internets fortschreiben. In teilweise zweifelhaften Allianzen[3] mit großen Konzernen wie *Google* oder dem Internet-Provider *Earthlink* haben in den USA einzelne Stadtverwaltungen wie San Francisco versucht, dem Problem mit einer Internet-Grundversorgung per WLAN–Funk[4] beizukommen. In Philadelphia setzten die Verantwortlichen zwar ebenfalls auf die Partnerschaft mit *Earthlink*, verzichteten aber auf eine kostenlose Grundversorgung zu Gunsten der Finanzierung von Computerkursen, Hardware und einer

3 Vgl. http://www.telepolis.de/r4/artikel/22/22454/1.html.

4 WLAN steht für „Wireless Local Area Network" und bezeichnet einen Standard für drahtlose Vernetzung von Computern in einem lizenzfreien Bereich des Funkspektrums. Die Sendestärke und damit die Reichweite einer einzelnen WLAN-Basisstation ist begrenzt, mehrere WLAN-Stationen können aber in einem „Mesh" genannten Funknetzwerk zusammengeschlossen werden.

Umsatzbeteiligung der Stadt sowie der Verpflichtung, eine günstige (ca. 10 Dollar pro Monat) Flatrate mit reduzierter Bandbreite anzubieten.[5]

Während eine vollständige Abdeckung des Gemeindegebiets mit einer kostenlosen Basis-Internetverbindung per Funknetz sicher alles andere als einfach zu realisieren und in ihren Wirkungen durchaus umstritten ist,[6] werden in Linz bis zum Kulturhauptstadtjahr 2009 erst einmal 130 WLAN-Basisstationen für kostenlosen Internetzugang errichtet.[7] In sämtlichen Bibliotheken, Volkshäusern, Jugendzentren und an viel frequentierten öffentlichen Orten ist mit einem WLAN-fähigen Laptop oder Handy der Einstieg ins Internet möglich. Der Vorschlag, diese Basisstationen in der Folge miteinander zu verknüpfen und eine vollständige Abdeckung zu erreichen, steht zwar im Raum, würde die notwendigen Begleitmaßnahmen wie erleichterten Zugang zu Hardware und die großteils bereits etablierten Computerschulungskurse keineswegs überflüssig machen.

4 Freies Wissen

Der Zugang zum Internet und zu Computern ist die absolute Grundvoraussetzung für jede weitere, darauf aufbauende Freiheit im Cyberspace und für die Entstehung einer digitalen „Wissensallmende". Die technische Möglichkeit und die Ausbildung, um sich im Netz bewegen zu können, nützt wenig, wenn die große Mehrheit der verfügbaren Inhalte nur gegen Entgelt zugänglich ist. Ein Umstand, der paradoxerweise im Bereich des stark öffentlich finanzierten Bildungssektors besonders weit verbreitet ist: Egal ob es um wissenschaftliche Texte, Kursunterlagen oder didaktische Materialen geht, sie alle sind wenn überhaupt dann nur gegen Geld online zugänglich.

In jüngerer Vergangenheit ist aber dank zwei Initiativen mit wiederum lokalem Ursprung auch dieses Feld in Bewegung gekommen: Open Access im Bereich wissenschaftlicher Veröffentlichungen und *OpenCourseWare* für den Bereich (universitärer) Lehrunterlagen. Beiden ist gemeinsam, dass sie die Bedeutung von spezifischen institutionellen Settings betonen, die freie Inhalte – abgesehen von ihrer Lizenzierung – erst dauerhaft verfüg- und nutzbar machen.

4.1 OpenCourseWare-Entwicklungsland Deutschland

OpenCourseWare (OCW) geht auf eine Initiative des *Massachusetts Institute of Technology (MIT)* zurück, das inzwischen zu mehr als 1 500 Kursen die Lehrunterlagen auf einem Online-Portal kostenlos und zur freien Weiterverwendung zur Verfügung stellt.[8] Dem Beispiel des *MIT* sind inzwischen größere und kleinere Universitäten

5 Vgl. http://www.telepolis.de/r4/artikel/22/22454/1.html.

6 Craig Settles verweist darauf, dass „diejenigen, die öffentliches WLAN am wenigsten benötigen, es am meisten nutzen" würden. Vgl. http://www.theregister.co.uk/2006/04/08/google_sf_muni_wifi/.

7 Vgl. http://www.linz.at/hotspot_Portal/hotspot.asp.

8 Vgl. http://ocw.mit.edu.

rund um den Globus gefolgt, von denen die Mehrheit auch im *OpenCourseWare Consortium* organisiert ist. Ziel dieser OCW-Initiativen ist die wechselseitige Verbesserung der Lehrqualität durch (interdisziplinären) Wissensaustausch und öffentliches Feedback. Neben dem mit *OCW* verbundenen Paradigmenwechsel für Lehrkräfte – didaktische Unterlagen zu teilen anstatt sie zu beschützen – ist es vor allem die institutionelle Betreuung und Wartung der Angebote, die erst eine sinnvolle Nutzung durch Dritte ermöglicht.

Im deutschsprachigen Raum gibt es abgesehen von ersten Gehversuchen keine nennenswerten Aktivitäten in diesem Bereich, die Webseite des OCW-Consortiums führt auch keine deutschsprachige Universität als Mitglied an. Genauso sieht es abseits der Universitätslandschaft aus: Die allerwenigsten Volkshochschulen und Erwachsenenlehranstalten versuchen, ihre Kursunterlagen frei verfügbar zu machen, obwohl das gerade ein Angebot an deren Zielgruppe weiterbildungshungriger Autodidakten wäre. Im Bereich der Schulen gibt es überhaupt noch keine Strategien und Initiativen für einen institutionellen Rahmen zum digitalen Austausch von Lehrmaterialien. Im Fall der Stadt Linz soll zumindest im Rahmen der städtischen Volkshochschule ein erster Schritt in Richtung *OpenCourseWare* unternommen werden. Vorschläge betreffend die Linzer Universität und den Grundschulsektor sind ob der komplizierteren politischen Zuständigkeiten erst in der Entwicklungsphase.

Klar ist lediglich, dass einzelne Schulen mit Aufbau und Betrieb eigener OpenCourseWare-Angebote überfordert wären und eine derartige Zersplitterung des Angebots auch nicht wünschenswert ist. Dabei ist die Aufgabe der Koordination und Erfassung von Lehrunterlagen mehrerer relativ selbstständiger Bildungseinrichtungen, wie sie sich kommunalen Schulausstattern stellt, dem Vorbild des MIT-OpenCourseWare-Programms gar nicht unähnlich: Auch dort wird ein institutioneller Rahmen für die Erfassung und Veröffentlichung von Materialien diversester Fachgebiete und relativ selbstständiger Lehrstühle als zentrale Dienstleistung organisiert.[9] Die zahlreichen privaten Homepages von Lehrerinnen und Lehrern mit Unterrichtsmaterialien zeigen jedenfalls, dass zumindest teilweise die Bereitschaft vorhanden ist, eigene Materialien anderen kostenfrei zur Verfügung zu stellen und auch ein Bedarf nach (inter-)disziplinärem Austausch besteht.

4.2 Open-Access-Vorreiterland Deutschland

Besser als im Bereich frei zugänglicher Lehrunterlagen sieht es dafür im Bereich frei zugänglichen, wissenschaftlichen Wissens aus, in dem diverse Open-Access-Initiativen versuchen, die herrschenden Regeln des wissenschaftlichen Publizierens zu verändern. Dem Titel der Initiativen entsprechend sollen die – ohnehin zum allergrößten Teil öffentlich finanzierten – Ergebnisse wissenschaftlicher Arbeit auch frei öffentlich zugänglich gemacht werden, damit andere schneller und einfacher wiederum auf

9 Das MIT-OCW-Projekt bietet unter http://ocw.mit.edu/OcwWeb/HowTo/index.htm Leitfäden für die verschiedenen Aspekte bei Aufbau und Betrieb einer OCW-Plattform an.

diesen Ergebnissen aufbauen können. So sollen die Potenziale des Internets gerade für wissenschaftlichen Diskurs in Form von Open-Access-Journalen oder zumindest von zusätzlicher Publikation der Ergebnisse auf Open-Access-Servern ausgeschöpft werden. Wie schon bei *OpenCourseWare* reicht auch bei Open Access die urheberrechtliche Freigabe der Inhalte allein nicht aus, es sind institutionelle Fragen (z. B. „Wer finanziert das Peer-Review-Verfahren bei einem Open-Access-Journal?", vgl. dazu Mruck et al. 2004) zu klären.

Auch hier war es eine lokale Initiative, nämlich die Budapester Open-Access-Initiative im Jahr 2001,[10] bei der mehrere Wissenschaftler in einer öffentlichen Erklärung den Stein ins Rollen gebracht hatten. Ihnen schlossen sich, im Zuge einer unterstützenden „Berliner Erklärung" im Jahr 2003,[11] inzwischen unter anderem auch die Deutsche Forschungsgemeinschaft (DFG) und die Max Planck Gesellschaft (MPG) an. Die weiterhin möglichen Beiträge von kommunalpolitischer Seite sind im Falle des entstehenden Open-Access-Publikationssystems zugegebenermaßen eher bescheiden und beschränken sich auf die Berücksichtigung im Rahmen von lokaler Forschungsförderung. In Linz soll dementsprechend ein Wissenschaftspreis spezielle für als Open Access frei verfügbare wissenschaftliche Arbeiten regionaler Forschungseinrichtungen vergeben werden.

5 Freie Kunst und Kultur

Während (die Unterstützung beim) Zugang zum Internet sowie zu Wissen im Bildungsbereich als Frage kommunaler Infrastruktur bereits in vielen Gemeinden als kommunalpolitisches Feld erkannt und bearbeitet wird, gibt es im verwandten Feld des kulturellen „Open Content" noch nahezu keinerlei Initiativen. Die allerwenigsten Gemeinden beschäftigen sich überhaupt mit Fragen wie der Lizenzierung von ihnen erstellter oder geförderter Inhalte. So stellen zwar viele Kommunen im Rahmen ihres Internetauftritts Panoramafotos und Bilder von Sehenswürdigkeiten zur Verfügung, zur Illustration einer entsprechenden Wikipedia-Seite dürfen sie deswegen aber noch lange nicht verwendet werden. Ähnlich sieht es mit anderen regelmäßig in Gemeinden produzierten Inhalten wie Jahrbüchern, Geschichtsbänden und touristischen Informationen aus.

5.1 Die Frage der Lizenzen

Dieses einfache Beispiel macht ein häufiges Dilemma im Umgang mit dem automatischen Urheberrechtsschutz von Werken und die Notwendigkeit einer Auseinandersetzung mit urheberrechtlichen Lizenzierungsfragen auch auf kommunaler Ebene deutlich. Denn in Europa ist jedes Werk – egal ob Foto, Film oder Text – nach seiner Erstellung prinzipiell einmal geschützt. Das führt zu der paradoxen Konsequenz,

10 http://www.soros.org/openaccess/read.shtml
11 http://oa.mpg.de/openaccessberlin/berlindeclaration.html

dass auch (kosten-)frei verfügbare Werke nicht ohne Zustimmung des Urhebers in anderen Kontexten verwendet werden dürfen. Schon gar nicht im Rahmen von freien Projekten wie der *Wikipedia*, die für ihre Inhalte vorschreibt, dass jeder sie verändern und weiterverwenden darf, solange auch die abgeleiteten Werke wieder frei verfügbar sind.[12] Die dafür notwendige Abklärung der Rechte ist aber in der Regel zu mühsam, umständlich oder teuer und stellt jedenfalls eine Barriere für die Nutzung dar, die oft auch von den Urhebern selbst so gar nicht intendiert war – Barrieren, die eine Art Kollateralschaden der (kommunal-)politischen Ignoranz gegenüber Urheberrechtsfragen im Zeitalter des Internets sind.

Genauso sieht es im Bereich des öffentlichen Förderwesens aus – hier allerdings auf allen politischen Ebenen. Obwohl oberstes Kriterium für den Erhalt von Subventionen das öffentliche Interesse an der geförderten Leistung ist, spielt die Frage der Lizenz für die Vergabe von Förderungen in den allermeisten Fällen keine Rolle. Ein Punkt, der für den Bereich der Kunst- und Kulturförderung in Linz bis zum Kulturhauptstadtjahr aufgegriffen werden soll ist folgender: Die Frage der Lizenz, unter der ein gefördertes Werk letztlich veröffentlicht werden wird, soll im Falle der Wahl einer freieren Lizenz mit einer erhöhten Förderwürdigkeit einhergehen. Gerade in Bereichen, wo ohnehin nicht kommerziell verwertbare Werke gefördert werden, stellt die mit der Verwendung einer Creative-Commons-Lizenz verbundene öffentliche Nutzbarkeit eine Steigerung des öffentlichen Mehrwerts dar.

5.2 Digital-kommunale Infrastruktur

Egal, ob für eigene Werke oder bei der Vergabe von Förderungen, allein die Berücksichtigung der Lizenzfrage auf kommunaler Ebene vergrößert den Pool an frei (weiter-)verwendbaren Werken und bringt Multiplikatoreffekte mit sich – ohne nennenswerte Kosten zu verursachen. Zusätzlich stellt sich die Frage, ob es auch in der digitalen „Arena" des Internets eine kommunale Verantwortung für die Schaffung von Kultur- und Diskursinfrastruktur gibt. Denn die immer umfangreicheren „kostenlosen" Angebote an virtuell-öffentlichen Räumen für die verschiedensten Zwecke – von Blog-Hosting bis hin zu Videoplattformen – haben in der Regel gleich mehrere Einschränkungen: Nutzbarkeit nur für spezifische Zwecke und vorbehaltlich geänderter Geschäftsbedingungen des Anbieters sowie häufig der Übergang der Urheberrechte von eingestellten Werken an den Betreiber.

In Linz wird seit Beginn des Jahres 2007 in den politischen Gremien der Vorschlag diskutiert, allen Einwohnern der Stadt sowie lokalen Kulturinitiativen Webspace zur nicht-kommerziellen Nutzung zur Verfügung zu stellen. So wie das Recht auf freie Meinungsäußerung und Versammlungsfreiheit im realen Raum nicht nur passive Duldung, sondern aktive Emöglichung und Schutz durch staatliche Stellen bedeutet, ist

12 Die *Wikipedia* stellt zu diesem Zweck ihre Inhalte unter die *GNU Free Documentation Licence (GFDL)*. Sie entspricht in ihren Wirkungen ungefähr der auch in diesem Band angeführten CC-BY-SA–Lizenz (siehe S. 563), ist allerdings nicht völlig kompatibel.

auch im virtuellen Raum die öffentliche Hand in die Pflicht zu nehmen. Dies umso mehr, da dessen Bedeutung als Sphäre des politisch-kulturellen Austauschs ständig zunimmt.

6 Fazit

In Rahmen ihrer Möglichkeiten können Kommunen gleich in mehrfacher Hinsicht wesentliche Beiträge für einen freieren Zugang zu digitalen Inhalten leisten. Ein Beitrag liegt schon allein in der Sensibilisierung innerhalb und außerhalb der kommunalpolitischen Einrichtungen für Fragen wie Lizenzierung von Inhalten und unintendierte Folgen eines unreflektierten Umgangs mit Urheberrechten im Zeitalter des Internets. Gerade durch die vielen unmittelbaren Bürgerschnittstellen auf lokaler Ebene liegt hier ein großes Potenzial für Multiplikator- und Schneeballeffekte.

Der Aufbau einer lokalen Infrastruktur als Basis zur Anbindung an und Nutzung der globalen digitalen Netze hingegen wäre ohnehin eine genuin kommunale Aufgabe. Ihre bislang äußerst spärliche Wahrnehmung ist abseits von mangelndem Verständnis mit Sicherheit auch eine Folge kommunaler Budgetnöte, die mit einer fast schon reflexhaften Abwehr des Aufbaus neuer kommunaler Leistungen einhergeht. Eine Abwehrhaltung, die in den vielen Bereichen ob der relativ geringen Kosten – insbesondere verglichen mit „klassischen" kommunalen Aufgaben – nicht auf Dauer aufrechtzuerhalten sein wird.

Literatur

Dobusch, L. und Forsterleitner, C. (Hrsg.) (2007), *Freie Netze. Freies Wissen. Ein Beitrag zum Kulturhauptstadtjahr Linz 2009*, Echomedia, Wien. http://www.freienetze.at/pdfs/fnfw(komplett).pdf [12. Jan 2007].

Hoegner, W. (2006), Das Projekt Limux – Freie Software für die Münchner Verwaltungsclients, *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 59–72. http://www.opensourcejahrbuch.de [12. Dez 2006].

Lutz, B. (2006), WIENUX und Open Office – die freiwillige Migration im Magistrat der Stadt Wien., *in* B. Lutterbeck, M. Bärwolff und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2006 – Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 73–82. http://www.opensourcejahrbuch.de [12. Jan 2007].

Mruck, K., Gradmann, S. und Mey, G. (2004), 'Open Access: Wissenschaft als Öffentliches Gut', *Forum Qualitative Sozialforschung* **5**(2). http://www.qualitative-research.net/fqs-texte/2-04/2-04mrucketal-d.pdf [12. Jan 2007].

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

Glossar

ACL (Access Control Lists, Zugriffskontrolllisten) werden zur Regelung von Zugriffsberechtigungen auf Ressourcen in IT-Systemen verwendet. Mit den einzelnen Einträgen in einer ACL kann für ein Objekt im System detailliert festgelegt werden, für welche Benutzer, Benutzergruppen oder Prozesse welche Art von Zugriff verboten oder erlaubt wird. · **Active Directory** ist ein zentraler Verzeichnisdienst zur Verwaltung eines Netzwerkes – hierzu zählen Benutzer- und Gruppenkennzeichen, Arbeitsplatzcomputer, Server und Drucker. · **Apache** ist ein weit verbreiteter Open-Source-Webserver, der ursprünglich aus dem NCSA-Webserver entstanden ist, unter der *Apache License* steht und federführend von der *Apache Software Foundation* betrieben wird. · **API** (Application Programming Interface – dt. Programmierschnittstelle) ist eine Menge von definierten Spezifikationen, die dem Programmierer den Zugriff auf Funktionen einer Software ermöglichen. · **ASP** (Application Service Providing) bezeichnet das Anbieten von Anwendungen, die Kunden eines Application Service Providers z. B. über das Internet zur Verfügung gestellt werden. Der Application Service Provider kümmert sich um die gesamte Administration, wie Datensicherung, das Einspielen von Patches usw. · **Backend** ist das Gegenstück zum Frontend, z. B. bei verteilten Anwendungen. Während das Frontend den für den Benutzer sichtbaren Teil der Anwendung darstellt, ist das Backend der für ihn unsichtbare, quasi im Hintergrund arbeitende Teil. Ein Bankautomat beispielsweise, an dem der Kunde Geld abhebt, stellt ein Frontend dar und der zentrale Server, der das Kundenkonto verwaltet, das Backend. · **Berliner Erklärung** ist die Kurzform für die „Berliner Erklärung über offenen Zugang zu wissenschaftlichem Wissen", die im Oktober 2003 auf einer Tagung der Max-Planck-Gesellschaft von deutschen und internationalen Forschungsorganisationen unterzeichnet worden ist. Sie ist ein wichtiger Meilenstein der Open-Access-Bewegung. · **Best Practice** ist ein Prozess oder eine Vorgehensweise, der/die sich in der Praxis mehrfach bewährt hat. · **Blackboard-System** dient in der künstlichen Intelligenz (KI) der Kommunikation der Programmstrukturen eines KI-Systems untereinander, indem es ihnen einen gemeinsam nutzbaren Speicherraum zur Verfügung stellt. · **Blog** ist die Kurzform von *Weblog*, einem öffentlichen Online-Journal, welches Beiträge chronologisch auflistet und die Möglichkeit bietet, die Einträge zu kommentieren und mit anderen Blogs zu verlinken. · **BSD-Lizenz** (Berkley Software Distribution License) ist eine von der *Open Source Initiative (OSI)* anerkannte Open–Source-Lizenz, die im Gegensatz zur *GPL* die Weitergabe von Software auch ohne den Quelltext billigt. Dadurch ist sie weniger restriktiv als die *GPL*. · **Bug-Tracker** bezeichnet ein System, zur Entdeckung, Verfolgung und Beseitigung von Programmfehlern. · **Business Intelligence** ist der Prozess, interne und externe Geschäftsinformationen zu sammeln und zu analysieren. Dies soll die Entscheidungsfindung der Leitungsebene unterstützen. · **Card Sorting** ist eine Technik um zu erforschen, wie Menschen bestimmte Gegenstände, Begriffe oder Einheiten gruppieren, damit Strukturen entwickelt werden können, welche die Wahrscheinlichkeit maxi-

mieren, Nutzer bestimmte Elemente finden zu lassen. Es ist einfach und billig durchführbar, ermöglicht herauszufinden, wie „echte Leute" Gruppierungen vornehmen würden, hilft beim Auffinden von schwer kategorisierbaren Objekten und identifiziert Technologien, die leicht missverstanden werden können. · **CEO** steht für *Chief Executive Officer*, in Deutschland der Unternehmensleiter, Geschäftsführer oder Vorstand. · **Cluster** ist ein Netzwerk von Computern, das nach außen hin wie ein einzelner Rechner agiert. Die interne parallele Bearbeitung von Aufgaben führt zu einer deutlich höheren Leistung. Cluster sind meist so organisiert, dass der Ausfall einzelner Computer nicht den Ausfall des gesamten Clusters nach sich zieht. · **CMS** (Content-Management-System) ist eine Applikation, die gemeinschaftliches Erstellen und Bearbeiten von elektronischen Dokumenten (Text, Multimedia) ermöglicht und organisiert. Wert gelegt wird auf eine möglichst einfache Bedienung, die keinerlei Programmierkenntnisse erfordert und z. B. ein schnelles Publizieren auf Webseiten erlaubt. · **Code-Review** ist ein iterativer Prozess, in dem geschriebener Quelltext von verschiedenen Personen validiert und auf Fehler hin untersucht wird. Dies soll eine hohe Qualität der Software gewährleisten. · **Colocation** bezeichnet das Anmieten von Raum in einem Rechenzentrum, um dort eigene Server aufzustellen, und von der vorhandenen Infrastruktur (z. B. Netzbandbreite) zu profitieren. · **Community** bezeichnet im Allgemeinen eine Gruppe von Individuen, die durch ein gemeinsames Element – meist sozialer Natur – verbunden sind, etwa ein gemeinsames Ziel oder eine gemeinsame Aufgabe. · **Content-Management-System** siehe CMS. · **Copyleft** bezeichnet eine Methode, die Freiheit der Nutzung, Modifikation und Weiterverbreitung einer Software zu sichern. So legt die *GPL* beispielsweise unter Verwendung des Konstrukts des Urheberrechts eines Autors fest, dass sämtliche Nutzungen und Modifikationen der Software erlaubt sind, eine Weiterverbreitung aber nur dann, wenn dabei die freie Verfügbarkeit des Quelltextes durch Erhalt der ursprünglichen Lizenz gewährleistet bleibt. · **Copyright** ist ein gesetzlicher Schutz im angloamerikanischen Rechtsraum, der Autoren, Künstler usw. vor unautorisierter Verbreitung ihrer Werke schützt. Im Unterschied zum deutschen Urheberrecht dient es hauptsächlich dazu, wirtschaftliche Investitionen zu schützen. Die Entscheidungs- und Verwertungsrechte können daher auch z. B. dem Verlag zustehen. · **Creative Commons** ist eine gemeinnützige Organisation, die verschiedene Lizenzen anbietet, mit denen Autoren allgemeine Nutzungsrechte an ihren Werken, wie z. B. Texten, Bildern, Musikstücken, einräumen können. Zu den einzelnen Abstufungen innerhalb der Lizenzen siehe auch S. 563. · **CRM** (Customer Relationship Management) bezeichnet die Verwaltung von Kundenbeziehungen eines Unternehmens. · **Cross-Kompilieren** beschreibt den Vorgang, bei welchem der Quellcode eines Programmes zur Verwendung auf einer anderen Plattform als der, auf welcher die Kompilierung stattfindet, übersetzt wird. · **CUPS** (Common UNIX Printing System) ist der Standard-Druck-Server auf vielen Linux-Distributionen und in *Mac OS X*. CUPS verwendet das *Internet Printing Protocol (IPP)*, um Druckaufträge, Warteschlangen und Netzwerkdrucker zu verwalten. · **CVS** (Concurrent Versions System) ist ein Softwaresystem zur Verwaltung von Quelltexten und anderen Ressourcen in Form von Dateien. Als sog. Versionskontrollsystem ermöglicht CVS das Speichern und Verwalten verschiedener Versionen einer Datei. Dies ermöglicht das gemeinsame Arbeiten mehrerer Personen an einer Datei und die Wiederherstellung älterer Versionen (ähnlich

der „Rückgängig-Funktion“ in Office-Programmen). · **Cyberspace** ist eine Wortschöpfung des Autors William Gibson, die sich auf virtuelle künstliche Netze bezieht. Das Präfix *Cyber* wird oft für Begriffe verwendet, die mit Computern und Netzen verbunden werden. *Cyberspace* wird mittlerweile als Synonym für das Internet gebraucht. · **DHCP** (Dynamic Host Configuration Protocol) ermöglicht mit Hilfe eines entsprechenden Servers die dynamische Zuweisung von IP-Adressen wie auch weiterer Konfigurationsparameter an Computer eines Netzwerks. · **Digital Divide** (engl. für: Digitale Spaltung) bezeichnet die Abkopplung einzelner Länder oder Gebiete von der globalen Informationsgesellschaft durch technologischen Rückstand. · **DMZ** (Demilitarized Zone) ist eine Art neutrale Zone eines Sicherheitskonzeptes in bzw. zwischen Netzwerken. Sie befindet sich meist zwischen einem privaten Netzwerk (z. B. Intranet eines Unternehmens) und einem öffentlichen Netz (z. B. Internet). · **DNS** (Domain Name Service) ist ein Dienst, der zwischen IP-Adressen und leichter verständlichen Domainnamen übersetzt. So steht z. B. die Domain opensourcejahrbuch.de für die IP-Adresse 130.149.24.71. · **Embedded-Systeme** sind Softwaresysteme, die direkt in die Hardware eines Gerätes eingebaut sind, etwa im Automobil- oder Entertainment-Bereich. · **ERP** (Enterprise Resource Planning) bezeichnet die Aufgabe, die in einem Unternehmen vorhandenen Ressourcen, wie z. B. Kapital, Betriebsmittel oder Personal, möglichst effizient für den betrieblichen Ablauf einzuplanen. · **Fachverfahren** sind Verfahren oder Abläufe, die durch Software unterstützt werden. Diese meist in Organisationen (z. B. Behörden) eingesetzten Verfahren resultieren aus sehr speziellen Aufgaben (z. B. Verwaltung der Hundesteuer), die sich nicht mit Standardsoftware bearbeiten lassen. Deshalb wird die entsprechende Software meist in Eigenentwicklung oder von einem Dienstleister erstellt. · **Fat-Client** ist ein in ein Netzwerk eingebundener Rechner, der (im Gegensatz zum Thin-Client) Programme und Daten lokal vorhält. · **forking** (engl. *fork* = Gabel, Gabelung) bezeichnet bei der Softwareentwicklung das Auf- oder Abspalten in eigenständige und sich parallel weiterentwickelte Projekte. · **Freeware** ist Software, für die keine Lizenzgebühren entrichtet werden müssen. Die Nutzung ist für die ausführbare Version fast uneingeschränkt möglich und die Weiterverbreitung ist auch erlaubt. Der Quellcode ist jedoch meistens nicht verfügbar. · **GCC** (GNU Compiler Collection) ist eine umfangreiche Suite zur Übersetzung von Programmcode unterschiedlichster Sprachen in Maschinensprache. · **GIMP** (GNU Image Manipulation Program) ist ein unter der *GPL* lizenziertes, umfangreiches Bildbearbeitungsprogramm. · **GNOME** (GNU Network Object Model Environment) ist eine sehr beliebte grafische Benutzeroberfläche für UNIX-Derivate, wie z. B. Linux. · **GNU** ist ein rekursives Akronym für „GNU's Not Unix“. Dieses steht für ein von Richard Stallman ins Leben gerufenes Projekt (1983 wurde die Idee erstmals in einer Newsgroup gepostet) zur Entwicklung freier Software. · **GNU General Public License** siehe GPL. · **GPL** (GNU General Public License) ist eine Open-Source-Lizenz, die von Richard Stallman geschaffen wurde, um Software samt Quelltext effektiv und nachhaltig zu einer Art öffentlichem Gut zu machen. Für eine unter der *GPL* stehende Software gibt es keinerlei Restriktionen bezüglich Nutzung, Modifikation und Weitergabe außer den Pflichten des Copylefts: Bei Weitergabe müssen Lizenz und Quelltext mitgegeben oder anderweitig zugänglich gemacht werden, und Änderungen am Quelltext müssen wiederum unter der *GPL* stehen. · **Hacker** ist eine Person,

die mit sportlichem Ehrgeiz die technischen Möglichkeiten von Hard- und Software zu ergründen und zu verwenden versucht. Daher kennt sich der Hacker gut mit Technik, insbesondere deren Stärken und Schwächen, aus. Hacken ist kreativer Umgang mit Technik. Teil der sog. Hacker-Philosophie ist, dass Informationen für jeden frei zugänglich sein sollen. Der Begriff „Hacker" ist urspr. wertneutral, wird aber von den Medien häufig i. Zshg. mit Computer-Kriminalität verwendet. · **Hebbsche Lernregel** ist eine nach Donald Hebb benannte Regel, die zuerst 1890 von William James formuliert wurde. Sie beschreibt den gegenseiten Einfluss benachbarter Neuronenverbände, die in zeitlicher Nähe zueinander aktiv sind, auf die Stuktur ihrer Verbindungen. · **Howto** ist eine Anleitung oder ein *Tutorial* zu einem bestimmten Thema. · **HTTP** (Hypertext Transfer Protocol) ist ein Protokoll zum Übertragen von Text-, Grafik-, Sound- oder anderen Dateien im *World Wide Web*. · **HTTPS** (Hypertext Transfer Protocol over Secure Socket Layer) bedeutet die mit SSL verschlüsselte Übertragung von HTTP-Paketen. · **Interoperabilität** beschreibt die Fähigkeit zur Zusammenarbeit von verschiedenen Komponenten (meist von unterschiedlichen Herstellern) eines heterogenen Gesamtsystems. Offene Standards begünstigen Interoperabilität. · **IRC** (Internet Relay Chat) ist ein Internetdienst, bei dem man sich in virtuellen Chaträumen, den sog. *channels*, mit anderen Personen zeitnah über Textnachrichten unterhalten kann. Man nennt dies auch „chatten". · **KDE** (K Desktop Environment) ist eine grafische Benutzer-Oberfläche v. a. für UNIX-Derivate, wie z. B. Linux. · **Kernel** ist der Kern eines Betriebssystems. Er koordiniert etwa die Verteilung der Hardware-Ressourcen an Applikationen und Zugriffe auf den Speicher. · **Knoppix** ist eine Linux-Distribution verbunden mit einer Live-CD oder -DVD, die ohne vorherige Installation auf einem Rechner von CD-ROM/DVD gestartet werden kann. Das von Klaus Knopper herausgegebene Knoppix ermöglicht ein Arbeiten mit vollwertiger grafischer Benutzeroberfläche inkl. Office-Programmen, ohne dass etwas an der Konfiguration des eingesetzten Rechners geändert werden muss. · **LAMP** steht als Akronym für eine Web-Plattform basierend auf den OSS-Komponenten Linux, *Apache* und *MySQL*, in Verbindung mit *Perl*, *PHP* und *Python*. · **LDAP** (Lightweight Directory Access Protocol) ist ein Protokoll zum Zugriff auf einen Verzeichnisdienst, in dem Daten unterschiedlicher Art in einer festgelegten Struktur gespeichert werden können. · **Load Balancing** (Lastverteilung) ist eine Technik, um Berechnungen oder Anfragen auf mehrere Systeme zu verteilen. · **Maintainer** bezeichnet eine leitende Rolle in Open-Source-Projekten, die dazu dient, Entwicklungsaufgaben zu koordinieren. · **Migration** bedeutet die koordinierte Umstellung von Systemen, etwa die Einführung von Linux auf Arbeitsplatz-Rechnern. · **Multicast** oder Gruppenruf bezeichnet in der Telekommunikation eine Nachrichtenübertragung von einem Punkt zu einer Gruppe (auch Mehrpunktverbindung genannt). Der Vorteil von Multicast besteht darin, dass gleichzeitig Nachrichten an mehrere Teilnehmer oder an eine geschlossene Teilnehmergruppe übertragen werden können, ohne dass sich beim Sender die Bandbreite mit der Zahl der Empfänger multipliziert. · **NDA** (Non Disclosure Agreement) ist eine im kommerziellen Softwarebereich verbreitete vertragliche Vereinbarung über die Geheimhaltung von Geschäftsgeheimnissen, zum Beispiel Quelltexten und Schnittstellenspezifikationen. · **Nerd** ist die meist abwertend verwendete Bezeichnung für einen fachlich herausragenden Menschen mit geringen sozialen Fähigkeiten. Positiv und aner-

kennend ist der Begriff dagegen unter Gleichgesinnten (Nerds) und bei Eigenbezeichnungen gemeint. · **Newsgroups** sind elektronische Foren im Internet, in denen man Diskussionen führen und Meinungen austauschen kann. · **NGO** (Non-Governmental Organization) ist eine Nichtregierungsorganisation und meint prinzipiell jeden auf gewisse Dauer organisierten Zusammenschluss von Menschen, der nicht gewinnorientiert, nicht von staatlichen Stellen organisiert bzw. abhängig und auf freiwilliger Basis aktiv ist. · **OASIS** (Organization for the Advancement of Structured Information Standards) ist eine internationale Non-Profit-Organisation, die sich mit der Weiterentwicklung von u. a. Dokumenten-Standards wie *OpenDocument* beschäftigt. · **OpenLDAP** ist eine OSS-Implementierung des *LDAP*. · **OpenOffice.org** ist eine freie Office-Suite, die aus dem kommerziellen *StarOffice* von *Sun Microsystems* hervorgegangen ist. · **Open Source Initiative** siehe *OSI*. · **Open-Source-Software** bezeichnet nach Definition der *OSI* Programme, deren Lizenzen bestimmen, dass der Quelltext offen vorliegt sowie verändert werden darf und dass es erlaubt ist, die Software beliebig zu kopieren, zu verbreiten und zu nutzen. · **OSI** (*Open Source Initiative*) ist eine unabhängige Organisation, die Lizenzen nach der Open-Source-Definition zertifiziert. Dazu muss die Softwarelizenz die freie Verfügbarkeit, Weitergabe und Veränderbarkeit der Software fordern. · **OSS** siehe Open-Source-Software. · **OS X** ist eine Version von *Mac OS*, Apples Betriebssystem für Macintosh-Computer. · **PageRank** ist ein Algorithmus der untereinander verlinkte Dokumente anhand der Struktur ihrer Verlinkung bewertet. Er wurde von Larry Page und Sergey Brin entwickelt. · **Patch** ist eine Änderung an einer Software, die für das Hinzufügen einer Funktion oder die Fehlerbereinigung, wie etwa das Schließen einer Sicherheitslücke, verwendet wird. · **Peer–Review** ist die fachliche Überprüfung von Artikeln durch gleichgestellte Fachleute. · **Perl** ist eine sehr flexible Skriptsprache, die vom Administrationswerkzeug bis zum dynamischen Webseiten-Generator in nahezu allen computertechnischen Einsatzbereichen verwendet wird. · **PHP** ist eine Skriptsprache, die v. a. für die dynamische Erstellug von Webseiten verwendet wird. · **Plug-ins** sind Zusatzmodule, die dazu dienen, Programme um bestimmte Funktionalitäten zu erweitern. · **proprietär** bezeichnet Software, Formate und Protokolle, die nicht „frei" sind, d. h. nicht (quell-)offen oder veränderbar. Nicht-standardisierte Entwicklungen werden ebenfalls als *proprietär* bezeichnet. · **Proxy** ist ein Dienst, der als „Mittler" fungiert. Er kann zwischen Rechnern, Netzwerken oder weiteren Diensten platziert werden und übernimmt dabei (je nach Auslegung) unterschiedliche Aufgaben. Beispielsweise kann ein Proxy eine Firewall (Nachrichten bzw. Pakete werden analysiert und ggf. ausgefiltert) oder einen Zwischenspeicher (*Cache* z. B. für Webseiten) darstellen. · **Python** ist eine objektorientierte Skriptsprache mit übersichtlicher Syntax. · **QPL** ist eine „Not-For-Profit" Softwarelizenz. Sie ist allgemein nicht kompatibel zur *GPL*, da man Veränderungen von QPL-lizenzierter Software nur in Form von Patches oder ähnlich deutlich gekennzeichnet verbreiten darf. Die *GPL* verbietet daher theoretisch die Verbindung (*linking*) von QPL-Software mit GPL-Software, manche Rechteinhaber wie die *Free Software Foundation* erteilen jedoch Sondergenehmigungen hierfür. · **Release** bezeichnet eine Softwareversion und den dazugehörigen Veröffentlichungstermin. · **Samba** ist OSS, mit der in einem Netzwerk Datei- und Druck-Dienste zur Verfügung gestellt werden, die sowohl von Linux- als auch von Windows-Clients genutzt werden können. · **Sendmail**

ist eines der am weitesten verbreiteten Mail-Transfer-Programme im Internet. Es steht seit der Version 8.9 unter der *Sendmail License*, vormals stand es unter der BSD-Lizenz. · **Server** ist ein Rechner, der über ein Netzwerk Dienste anbietet, oder ein Programm, mit dem ein solcher Dienst bereitgestellt wird. · **Skript** ist genau wie ein Programm eine Reihe von Anweisungen für den Computer, die allerdings nicht zuvor kompiliert werden müssen. Bekannte Skriptsprachen sind *Perl*, *PHP*, *bash*, *Ruby*, *Python*, *Tcl* und *Applescript*. · **Smartcard** ist eine Chipkarte mit eingebautem Kryptoprozessor. Ein bekanntes Beispiel für eine Smartcard ist die Patienten-Chipkarte der Krankenkassen. · **Software-Stack** ist eine Menge von aufeinander aufbauenden Softwarekomponenten in einem größeren System. So bilden etwa Linux, *Apache*, *MySQL* und *Perl* den LAMP-Stack. · **SQL** (Structured Query Language) ist eine (teilweise genormte) strukturierte Abfragesprache zur Kommunikation mit Datenbanken. · **SSL** (Secure Socket Layer) ist ein Protokoll zum Verschlüsseln von Internet-Verbindungen. Es wird u. a. von Webbrowsern unterstützt und gewährleistet eine abgesicherte Übertragung von Nachrichten über das Internet. · **Stack** siehe Software-Stack. · **Structured Query Language** siehe SQL. · **TCO** (Total Cost of Ownership) umfasst die Gesamtbetriebskosten der Infrastruktur im laufenden Betrieb. Dabei werden nicht nur die budgetierten Kosten der IT-Abteilung (Hardware, Software, Personalkosten für Anschaffung und Maintenance), sondern auch die indirekt anfallenden Kosten bei den Endanwendern betrachtet. Üblich ist dabei ein Betrachtungszeitraum von einem Jahr. · **Thin-Client** ist ein Computer, der in einem Netzwerk als Endgerät (Terminal) dient. Benötigte Programme werden über das Netzwerk geladen, und Daten werden auf einem entfernten Speicher abgelegt (vgl. *Fat-Client*). · **Thread** (engl.: Faden, Strang) ist in der Informatik ein Programmteil, der zeitlich unabhängig von anderen Programmteilen abläuft. Im Zusammenhang mit E-Mails und Newsartikeln werden alle Beiträge, die sich auf eine Ausgangsnachricht bzw. einen Artikel beziehen, ebenfalls als Thread bezeichnet. · **Total Cost of Ownership** siehe TCO. · **Usability** bezeichnet die Benutzerfreundlichkeit einer Applikation. · **Usenet** war ein UUCP-basiertes Netzwerk in den 1980er und frühen 1990er Jahren für den regen Austausch in Diskussionsforen, sog. Newsgroups. Das UUCP-Verfahren (*Unix to Unix CoPy*) verlor durch den Boom des Internet stark an Bedeutung. · **Verzeichnisdienst** speichert Daten unterschiedlicher Art in einer festgelegten Struktur. Ein Verzeichnisdienst wie *OpenLDAP* stellt diese in einer IT-Infrastruktur zur Verfügung. · **VMS** Die Firma *Digital Equipment Corporation (DEC)* produzierte Maschinen zur elektronischen Datenverarbeitung, die mit dem Betriebssystem *VMS (Virtual Memory System)* ausgestattet waren. „Vater" dieses Betriebssystems war Dave Cutler, der später die Architektur von *MS Windows NT* gestaltete. · **Weblog** siehe Blog. · **Wiki** bezeichnet das Konzept von Websites, auf denen Besucher Texte nicht nur lesen, sondern auch unkompliziert bearbeiten können. Bekanntestes Beispiel ist die *Wikipedia*. · **XDMCP** (X Display Manager Control Protocol) ist ein Netzwerkprotokoll, dass der Kommunikation zwischen einem X-Terminal und einem X-Server dient. · **XML** (Extensible Markup Language) ist eine standardisierte Technologie zur Erstellung von Austauschsprachen und -formaten.

Stichwortverzeichnis

3D-Drucker 116
3D-Modell 116

A
ACL 531
ACP 403
Active Directory 531
affiliate marketing 92
Agendasetting 490
AIX 135
Allgemeinbildung 345
Allgemeinheit 422
Allman, Eric 28
Allmende 472
Alltagsalgorithmus 348
Altruismus 171
Amarok 295
Amazon 137, 168
American Scientist 406
Analphabeten 393
Analyse 22, 54, 70, 75, 92, 178, 185–192, 197–207, 210, 216, 231, 235, 293, 353–356, 424, 442, 483, 484, 502, 515
  Portfolio-~ 201
Anbietermonopol 524
Anbietungspflicht 433
Aneignung
  soziale ~ 486
  soziokulturelle ~ 60, 61
Anforderungsanalyse 488
Antiterrordatei 465
Apache 18, 19, 48, 210, 241, 274, 531
Apache Tomcat 370
API 129, 531
Apple 29
Application-Server 198
Application-Service-Providing-Software . 197
Arbeit
  ~sexternalisierung 107
  ~skraft 106
  Eigen~ 115
  Erwerbs~ 353
Arbeitsteilung 89, 95
Arcangel, Cory 330
Archan, Sabine 344
Archivserver 420
Arpanet 27, 28, 49, 475
ArXiv 166, 398, 401
ASCAP 450
ASCII 29, 407
ASP 155, 201, 531
  Lösungen 198, 200
Astaro Security Linux 382
ATM 242
Attestation 303, 306, 307
Außenpolitik 239
Aufblühen der Wissenschaft 108
Auslandsvertretung 240
Ausschluss
  ~ von Nutzern 485
  ~ von Nutzungen 485
Ausschreibung 227
Auswärtiges Amt 232, 239
Authorisation 245
AutoCAD 387
Autodesk 141
Autodidakt 527
Automation 106
Ax, Christine 114

B
Backend 531
Ballmer, Steve 19, 134, 226–230
Bandbreite 277, 278, 284
Barksdale, Jim 47
Basarstil 75, 488
BASF 93
Basisdemokratie 489
Basisqualifikation 344
Bauernprivileg 105
Bearbeitungsgemeinschaft 417
Bearbeitungsverfahren 115
Behörde 224–235

Bell Labs 27, 28
Benchmarking 284
benefit sharing 477
Benjamin, Walter 329
Benkler, Yochai 93, 103, 113
Benutzerfreundlichkeit 198, 202, 229
Benutzerschnittstelle 246
Benutzungsfreundlichkeit ..... *siehe* Usability
Bergmann, Frithjof 110, 114
Berkeley 28
Berlin 228–238
Berliner Erklärung 430, 528, 531
Berlios 278
Berners-Lee, Tim 18, 38, 41, 43, 141
Bernstein, Basil 355
Best Practice 531
Betriebssystem .. 198, 202, 205, 207, 224–237
~-Image 280
Bibliometrie 397
Bildung 388
~sbereich 528
~sbetrieb 366
~sministerium 373
~ssystem 111
Schul-~ 173
Billigprodukte 106
BIND 18, 28
binding 306
Biodiversität 119
Biogene Reststoffe 108
Bioinformatik 41, 75, 84
Biolinux 103
Biomasse 108
BitLocker 312
Blackboard-System 175, 531
BLAST 75
Blender 336
Blog 529, 531
Bot-Netz 299
Botschaft 240
Braungart, Michael 117
Brecht, Berthold 484
Breitenelemente 344
Bridging 267
British Medical Journal 397
BSD 315
~-Lizenz 157, 319, 414, 531
~-Varianten 30
Free~ 30
Net~ ..... *siehe* NetBSD
Open~ 30
BSI 241
Bug 219
Bug-Tracker 243, 531
Bundesinnenministerium 482
Bürger-Beteiligung 482, 487
Bürgerkarte 371
Bush, Vannevar 37
Business
~ Intelligence 531
~ Process Reeingeneering 90
~ Software Alliance 146

C

car
~ pooling 470
~ sharing 470
Card Sorting 290, 531
Castells, Manuel 52
CC 396, 461, 523, 529, 532
~-Lizenz 417, 431, 448, 463, 529
~-Projekt 448
CCSD 398
Cell Science 431
Cendent vs. Amazon 137
CEO 169
CERN 409
cgd 322
Champy, Hammer 90
charismathics 287
Cheating 63
Clark, David D. 485
Client 223–237
Cluster 198, 200, 532
CMS 198, 204, 206, 207, 245, 371, 532
CO_2 108
Code
~ is law 485
~-Review 532
~-Savvy-Nutzer 293
geschlossener ~ 355
integrativer ~ 355
Colocation 318, 532
Commons 93, 95
commons-based peer production 75
Community 190, 291–297, 506, 532
~-Management 191
KDE-~ 297
Multipurpose ~ Center 111
OSS-~ 191
Usability-~ 291
Compartment 309
Computer
~ im Unterricht 355

~ literacy 358
~kabinett (im Unterricht) 360
~spiel 26, 63, 64, 67
~technologie 66
~theorie 25
Computer Science Teachers Association . 346
Concord Fallacy 189
Conquistadores 119
Content-Management 201
~-System ..... *siehe* CMS
Coopetition 48
Copyleft . 39, 40, 75, 77, 78, 83, 415, 423, 532
Copyright 18, 170, 532
Creative Commons ..... *siehe* CC
CRM 532
CRM-/SRM-System 197, 200, 201, 204
CRM/SRM 205
Cross-Kompilieren 316, 532
CUPS 532
Curricula 355
CVS 532
Cyberspace 533

D

Dahl, Robert A. 482
Dari 380, 387
DARPA 157
Datenbank 198–207, 220
~server 241
Datenschutz 470
DB2 152
DDD 409
DDR 393
Debian 224, 242, 379
~-Entwickler 243
~-Translation-Projekt 243
Deliberation 484
Dell 48
Demokratische Kontrolle 41
Demokratische Mitbestimmung 482
Demokratisierung 34, 42
DEMOS-Projekt 487
Denial-of-Service-Angriff 299
Desktop-Management 285
Deutsche Forschungsgemeinschaft *siehe* DFG
Deutsche Nationalbibliothek .. 398, 402, 406
Deutscher Akademischer Austauschdienst 379
Dezentralisierung 106
DFG 400, 406, 528
DHCP 533
di Castelbianco, Colletta 110
di Lucca, Bagni 118
DiBona, Chris 370
Didi, H.-J. 345
digg.com 489
Digital
~ Divide 408, 484, 525, 533
~ Peer Publishing License 423
~ Rights Management ..... *siehe* DRM
~e Modernisierung 22
~e Spaltung ..... *siehe* Digital Divide
~es Netz 530
~isierung 26, 27, 45, 50
DiMeB 343
DINI 397, 400, 406
Diskurs
~fähigkeit 484
~ive Öffentlichkeit 484
herrschaftsfreier ~ 484
Display Manager 282
Dissertation 402, 406
DMZ 533
DNS 533
DOAJ 402
Dokumenten-Management-System 197–204, 247
Dokumentenformat 213
Doom 48
Dotcom 18, 48, 167
~-Blase 167
Dougherty, Dale 490
DPPL 421
DRBD 246
Dreamweaver 387
Drehbuch 355
DRM 5, 311, 419, 427–436, 511
Dublin Core 403

E

E-Administration 372
E-Card 371
E-Demokratie 482–491
E-Government 156
E-Portfolio 375
eBay 154, 168
Economy of scale 106
Edubuntu 362
EduKnoppix 362, 369
eduMoodle 374
EDV-Chef 224–229
EDV-Kustode 369
Effekt
Forking-Thread-~ 189
Multiplikator-~ 530

Schneeball-~ 530
Wissensspillover~ 188, 191
Effizienz 90, 95, 186, 187
Kosten~ 90
eFit 370
Eigensteuerung 355
Eigentätigkeit 107
Eigentum
geistiges ~ 21, 36, 39, 40, 50, 52, 95, 462
Eigentumsrecht 80, 81
geistiges ~ 79, 212
eLearning-Cluster 373
Electronic commerce 90
Eli Lilly 93
Elsevier Science 403, 409, 411
Emacs 35
embedded systems 301, 315
Embedded-Systeme . *siehe* embedded systems
EMSCB 302
Emulation 264
End-to-End
~-Debatte 491
~-Design 485
Endorsement Key 303
Entgeltsicherung 421
Entwicklergemeinschaft 215
Entwicklung
~sprojekt 373
~sstil 186
~ssystem 198, 200, 201, 207
Doppel~ 190
ko-evolutionäre ~ 481, 487, 489
Epoche
digitale ~ 21, 37, 51
industrielle ~ 40
ERP 201, 533
ERP-System 197, 198, 200, 204, 205
Ersturheber 416
Erstveröffentlichungsrecht 433
Erwachsenenlehranstalten 527
Erziehung 464
eSV 371
Ettrich, Matthias 489
EU 77, 115, 230, 240, 244, 372, 375, 512
European Computer Driving Licence ... 370
Evolution 107
Evolutionsstrategien 488
Executive Committee 318
Exim 241
EZB Regensburg 402

F

Fach
~anwendung 225, 231, 242
~gesellschaft 399, 405, 406
~verfahren 225–235, 533
Fat-Client 274, 533
Feedback 172
Fenstermanager 282, 283
file flag 322
file sharing 464, 471
~-Protokoll 62
Firefox 224, 233, 242, 275
Firewall 278
Fleissner, Peter 109
Förder
öffentliches ~wesen 529
~würdigkeit 529
Fordismus 20, 22, 25, 45
forking 190, 191, 533
Forschung 231
~sförderung 528
sozialwissenschaftliche ~ 483
Weblog-~ 484
FOSS 523
Four Rules of Use 450
Fraunhofer-Institut 195, 230
Free Software Foundation ..... *siehe* FSF
free speech 477
FreeBSD 317
Freedesktop.org 275
FreeNX 273–288
Freeware 533
Freie
~ Automaten 109
~ Baupläne 109
~ Designs 109
~ Meinungsäußerung 529
~ Netze 525–526
~s Wissen 526–528
Freie/Open-Source-Software .... *siehe* FOSS
Freiheit 1
Fremdsteuerung 355
Frieden 240
Frontalunterricht 360
Fry, Ben 336
FSF 35, 39, 134, 523
FTP 317
Funkverbindungen 240

G

Game modding 64
GCC 332, 533
Gehirnwäsche 244

GEMA 418, 444
Gemeinbesitz 18
Gemeinde 528
Gemeineigentum 79
  geistiges ~ 39
General Electric 27
Genmaterial 104
Geschäftsmodell 285, 431
Gesellschaft 21
  Informations-~ 353
  konsumistische ~ 106
Gestaltungsentscheidung
  technische ~ 481, 490
Gestaltungsmöglichkeiten
  ~ demokratische 482
Gewährleistung 496
GFDL 529
Gibson, William 60
GIMP 292, 293, 359, 533
Give Get Nation 115
Globale
  ~ Dörfer 109
  ~ Urbanisierung 110
Globalisierung 61, 110
GNOME .. 143, 218, 276, 277, 289, 291, 533
GNU 2, 29, 43, 533
  ~ Free Documentation Licencse .. *siehe* GFDL
  ~-Bewegung 43
  ~-Projekt 35
  ~-System 133
  GPL . 29, 39, 40, 43, 150, 190, 229, 277, 313, 415, 419, 462, 511–521, 533
  LGPL 190, 414
Golden Road 430, 431
Goldene Nica 523
Google 217, 374, 404
Gorz, André 354
GPL ..... *siehe* GNU GPL
GPRS 278
Grant-back-Bestimmung 83
Granularität 94, 98
Green Road 430, 431
Groenemeyer, Marianne 106
Groklaw 230
Groupware 242
Groupware-Applikation 198
Grundschulsektor 527
Gruppe
  altersheterogene ~ 360
  leistungsheterogene ~ 360
Güter
  öffentliche ~ 187
  digitale ~ 62
  immaterielle ~ 62

H

Hacker 24, 25, 27, 35, 39, 46, 67, 533
Haftpflicht
  ~versicherung 501
  Produkt~ 496, 500, 502, 505, 507
  Produzenten~ 496
Haftung 496
  ~srisiko 422, 496
  Produkt~ 496, 502, 508
HAL 398–402
Halloween Papers 19
Handelsakademien 369
Handwerk 114
Hardware 229–236
  ~architektur 264
  ~komponente 264
  ~unternehmen als OSS-Produzenten .. 154
  physikalische ~ 264
  x86-~ 268
Harnad, Stevan 406
HBCI 310
Herat 380
Herrschaftsfreier Diskurs 484
Himanen, Pekka 354
Hoecker, Beate 484
Hoegner, Wilhelm 224–230
Holzer, Derek 334
Homo
  ~ Carbonis 108
  ~ Siliconis 108
Homöostase 108
Horizontalqualifikation 344
Howto 330, 534
HP 153
HP OpenView 241
HTTP 18, 31, 534
HTTPS 534
Hughes, David J. 28
Human-interface guidelines 297
Hypervisor 265

I

IANA 278
IBM ... 48, 135, 153, 214, 215, 226, 230, 291
IceWM 282
ideeller Lohn 489
IDS 300

Impact 397, 399, 401, 403, 408, 420
Indien 111
Industrialisierung 36, 37, 52
Industrie 21
Industriegesellschaft 107
industrielle Ordnung 21
Informatikerausbildung 221
Information 170
~ Sharing 462
~ Super Highway 466
~sdienstleister 406, 410
~sfreiheit 430
~sgesellschaft 172, 472
~sgut 165
~squellen 421
~stechnologie 273
~sversorgung 395, 396, 410
Informationelle Selbstbestimmung 352
Infrastruktur 111, 225, 229, 231, 486
Innocentive 92–95
Innovation 61, 70, 88, 91, 95, 219, 231
~saktivität 185
~sdruck 21
strukturelle ~ 19
Innovative Medicines Initiative 77
Insourcing 244
Integration in die Wirtssysteme 270
Intel 41, 48, 215
Interaktion 89
~skompetenz 100
~splattform 94
Internet 18, 27, 28, 31, 42, 44, 50, 76, 90
~ als neues Massenmedium 486
~ im Unterricht 355
~-Community 97
~-Nutzung 484
~archiv 440
~foren 484
~infrastruktur 67
demokratische Potenziale des ~ ... 491
freier Zugang zum ~ 491
Nutzungsweisen des ~ 485
Interoperabilität 127, 217, 244, 534
Intranet 232, 240
Intrusion-Detection-System ..... *siehe* IDS
IoPP 409, 411
IP 387
IPFilter 323
IPsec 240
IPv6 267
IPX 18
IRC 326, 534
Ishii, Maki 439
ISO-Norm 13407 290
Isolation 308
IT
~-Aufbauprojekt 378
~-Budget 239
~-Center Kabul ..... *siehe* ITCK
~-Dienstleistungen 128
~-Entwicklungen 239
~-Fachbereiche 240
~-Fachexpertise 380
~-Haushalt 240
~-Infrastruktur 243
~-Landschaft 231, 234, 244
~-Personal 244
~-Referat 240
~-Schlüsselqualifikationen 343
~-Servicezentren 240
~-Sicherheit 240
~-Strategie 229, 240
~-Versorgung 379
~-gestützte Fachverfahren 247
ITCK 379, 380, 382, 387
IuK 406

J

Jändel, Adolf 111
Jakubowski, Marcin 118
Java 225, 226
JBoss 242
jBPM 247
JISC 400
jMax 334
journal crisis 428
Joy, Bill 29
JULU 332
Just in time 110

K

Kabul 379
Kapitalvorschuss 106
Kassensystem 277
Katz, Jon 18
KDE . 218, 224–229, 244, 276–284, 289–295, 325, 347, 534
~-Community 297
Kelly, Kevin 107
Kerberos 286
Kernel 274, 280–282, 284, 316, 534
Linux-~ 229, 274, 280–282, 284
Kernel Security Level 322
keylogger 300

King, Patrick 335
Kirchbach,Steiermark 110
Kitesurfing 96
Kivio 295
Klafki, Wolfgang 345
Kleinbetriebe 112, 114
Knoppix 361, 368, 534
Kommunalpolitik 523
Kommune 228, 285, 528, 530
Kommunikation 483
Kommunismus 453–456
Kompatibilität 31
Komplexe Systeme 107
Koncraft 113
Konkurrenzkampf am Zulieferermarkt .. 115
Kontact 295
Kooperation 188, 227
  dezentrale ~ 488
  globale ~ 19
Kopierschutz 61, 170
Kosten 62, 229, 235–237
  ~spareffekt 189
  ~senkung 231
  ~teilung 82
  ~unterschiede 187
  Administrations~ 218
  Anschaffungs~ 215
  Betriebs~ 218, 231, 235
  Lizenz~ 206, 227–234
  Lohn~ 216
  Opportunitäts~ 187
  Produktions~ 89
  Switching-~ 125, 128–130
  Transaktions~ 74–80, 89, 90, 502
  Umstellungs~ 216
Kriterienkatalog 347
Kultur
  ~ der Partizipation 113
  ~-Gemeinschaft 353
  ~en früherer Zeit 119
  digitale ~ 20, 27
Kulturhauptstadt 523
  ~jahr 524, 529
Künstliche Knappheit 116
Kuroo 295

L

L4 308
LAM 255
LAMP 274, 370, 534
Langzeitarchivierung 398, 400, 410, 430
Laptop 146, 232, 526
Lastverteilung 241
LDAP 249, 285, 286, 387, 534
  ~ Account Manager ..... *siehe* LAM
  ~-Server 245
Lehr
  ~mittelfreiheit 358, 393
  ~unterlagen 527
Lehramtsstudierende 346
Lern
  ~kultur 366
  ~managementsysteme 367, 371
  ~materialien 374
  ~medien-Ausstattung 360
  ~plattform 367
  ~regel, Hebbsche 178
  ~statt 111
Lernen
  fächerübergreifendes ~ 355
  selbstreguliertes ~ 355
Lernender Stadtteil 111
Lernendes Unternehmen 118
Lessig, Lawrence 22, 41, 70, 485
Leuchtturm 524, 525
Libre Graphics Conference 293
LiMux 223–230, 524
Linus' Law 175
Linux . 1, 7, 19–51, 65, 67, 126–130, 166, 206, 209–222, 224–238, 274, 352, 380, 385
  ~ Backoffice 251
  ~ Router Project 219
  ~ Solution Group ..... *siehe* LiSoG
  ~ Standard Base 150
  ~ Terminal Server Project .. *siehe* LTSP
  ~ Terminalserver 273–288
  ~-Derviate 266
  ~-Desktop 382
  ~-HA-Cluster 246
  ~-Installation 267
  ~-Kenntnisse 267
  ~-Kernel 229, 274, 280–282, 284
  ~-Umgebung 267
  ~hotel 244
  ~migration 248
Linz 523, 527–529
LiSoG 281, 284, 285
Lissabonziele 371
Lizenz 75, 77, 79, 80, 96, 226
  ~bedingungen 189, 211, 447
  ~beschränkung 212
  ~bestimmungen 126
  ~entwicklung 80

~erlös 81
~gebühr 212
~geber 79–81
~ierung 69, 75, 78, 95
~kosten 227–234
~nehmer 80, 81
~standard 78
~verfasser 80
~vertrag 414, 417, 418
~vertragsgestaltung 417
BSD-ähnliche ~ 83
BSD-~ 531
Click-Wrap-~ 77
Copyleft-ähnliche ~ 83
Copyleft-~ 81
cross-licensing 516
GPL ..... *siehe* GNU GPL
Noten~ 449
Open-Source-ähnliche ~ 84
Open-Source-~ .. 75, 77, 79, 80, 82, 83, 97, 497, 498, 507
OSS-~ 78, 79, 83, 189, 500, 507
proprietäre ~ 83
Software-~ 80
LMS 371
Load Balancing 246, 534
Lobby 234
~ing 223–232
~ismus 184
~ist 234
Lock-in 128, 130, 135, 156
Lock-in-Effekt 211, 216, 221, 222
Logical Volume Manager 246
Lovins, Amory 117
LTSP 281, 361
Lutterbeck, Bernd 486
LVM 246
Lyle, John 107, 117

M

Mac OS X 535
Mackwitz, Hanswerner 117
Mailserver 241
Mainframes 24, 25, 67
Maintainer 534
MAKE 116
Malware 63
man pages 317
Management
~-Framework für Thin-Clients 285
Planetares ~ 108
Many-to-Many-Struktur 485
Margulis, Lynn 108
Markt
~anteil 228
~modell 168
~nachfrage 429
~ökonomie 166
~risiko 89
~wirtschaft 117, 159
Maschine 265
Lebende ~ 107, 117
virtuelle ~ 265
Massachusetts Institute of Technology .. *siehe* MIT
Materielle Absicherung 109
Materielle Selbstversorgung 109
Mauss, Marcel 68
Max Planck Gesellschaft 528
Max/MSP 333
McDonough, William 117
McLuhan, Marshall 107, 329
Medien
~ecke 360
~kompetenz 363
~kunst 328
~rauschen 484
Medknow Publications 408
Meritokratie 489
Mertens, Dieter 344
Merton, Robert K. 31
Merz, Markus 112
Metadaten 398, 400, 403–405
Microsoft . 19, 29, 40, 49, 130, 186, 205, 214, 217, 218, 223–238, 240, 264, 291
~ Excel 130, 224
~ Exchange 241
~ Internet Explorer 237, 241
~ Mail 240
~ Office 241, 380
~ Outlook 205
~ Outlook Express 241
~ Windows 227, 240, 241, 379, 380
~ Word 130, 226, 407
~-Modell 133–160
Middleware 138, 231, 242, 310
Migrantenanteil 361
Migration .. 203, 205, 207, 224–236, 249, 534
~skosten 253
~sstrategien 130
Live-~ 267, 271
Mikroprozessor 28
Ministerium 233
Außen~ 232

Innen~ 228, 233
Wirtschafts~ 233
MIT ... 25–28, 35, 41, 94, 115, 275, 526, 527
Modularität 417, 423
Monopol 74, 75, 81, 83
~ist 225
~renten 186
~stellung 428
Moodle 373
Moravec, Hans 108
Motivation ... 51, 91, 94, 95, 98, 99, 188, 190, 191, 215, 217, 502
Motorola 41
Mozilla 370
Public License 414
mSQL 28
mtree 322
Multiboot-Client 243
Multicast 534
~ TFTP 277
Multics 27
Multiplikatoreffekt 529, 530
München 206, 223–237, 524, 525
Mundie, Craig 19, 40
MySQL 151, 274

N

n4education 373
Nachbaumöglichkeit 105
NATO 379
NDA 534
Nelson, Rick 108
Nelson, Ted 39, 141
Nerd 534
Nestlé 93
NetBEUI 18
NetBSD 30, 190, 315–326
~ Core Team 317
~ Foundation 318
~ Security Alert Team 321
~ Security Officer 321
~ Sicherheitsmechanismen 322
Netscape 18, 47
Netzwerk 88, 90
~effekt 126, 127, 129, 524
~komponenten 241
~unterstützung 267
new economy 18
New York 240
Newsgroup 535
Newton, Isaac 59
NGO 535
NGSCB 312
NLPOS 277, 285
NoMachine 256, 277
non-rivalry 170
Norddeutsche Affinerie AG 249
Not-invented-here-Syndrom 99
Notenmanuskript 442
Novartis 93
Novell 48, 230–237, 284, 285, 372, 499
~ Bordermanager 372
~ Groupwise 372
~ Open Enterprise Server 372
Novell Linux Point of Service .. *siehe* NLPOS
Nutzerkontakt 190
Nutzungsrecht 414–416, 422, 430, 520
~einräumung 520
NX 277–278

O

O'Reilly, Tim 115, 488, 490
OASIS 213, 535
Ocean Arks 117
offene Jugendarbeit 361
öffentliche Institutionen 214, 220
öffentlicher
~ Bereich 215, 228
~ Dienst 211, 228
~ Sektor 211, 222, 230
Office 130, 202, 205–207, 213
~-Produkt 200
~-Suite 224, 535
Microsoft ~ 224, 237
Ökonomie 189
Gaben~ 68, 70
Geschenk~ 132
Netzwerk~ 126, 127
Traditionelle ~ 168
Ökosysteme 108
Online
~-Befragungen 366
~banking 309
Open Access ... 166, 171, 395–411, 420–423, 427–436, 523, 526–528
~-Ansatz 421
~-Archivierung 431
~-Community 168, 171
~-Dienstleister 404–405
~-Gemeinde 178
~-Geschäftsmodell . 397, 403, 405–408, 410
~-Initiative 528
~-Journal 528

~-Lizenz 422
~-Philosophie 173
~-Publikation 173, 422, 430
~-Publikationssystem 528
~-Server 528
~-Verlag 398, 399, 402, 405
~-Zeitschrift 397–410, 431–436
Open Archives Initiative 400, 404
Data Provider 403–408
OAI-PMH 400
Service Provider 404
Open Computing 126, 129
Open Content 416–419, 431, 443, 528
~-Ansatz 442
~-Gedanke 440
~-Gemeinde 417
~-Konzept 450
~-Lizenz 418, 421
~-Prinzip 451
Open Document Format 213, 234
Open Fiction 453
Open Source 1–545
~ Best Practice Award 250
~ Definition 132
~ Gipfel 370
~ Initiative 3
~ als Idee 491
~ der Güter 116
~ in der Biotechnologie 105
~ in der materiellen Produktion ... 103
~ und ökologische Technologien .. 117
~-Ansatz 79
~-Anwendung 210
~-Anwendungsfeld 205
~-Betriebssystem 197, 315
~-Bewegung 131–164
~-Car 112
~-Community 347, 487
~-Datenbank 285
~-Dienstleistungen 52
~-Dorf 118
~-Ecology 118
~-Entwicklung 172
~-Entwicklungsmodell 218
~-Entwicklungsprojekt 82, 502
~-Gemeinschaft 65, 67
~-Geschäftmodelle 149
~-Grundsatzprogamm 134
~-Informationsgut 166
~-Lösung 196, 231
~-Lizenz 414, 415
~-Methode 75
~-Phänomen 68, 70
~-Praxis .. 26–28, 34, 39, 43, 47, 51–53
~-Prinzip 76, 77, 80, 81, 84, 105
~-Produkt 52, 128, 129, 196, 207
~-Produktionskette 416
~-Produktmarketing 158
~-Programm 230, 414
~-Programmierer 69, 95
~-Projekt 19, 30, 67, 126, 203–205, 207, 218, 221, 282
~-Prozess 30
~-Software 204, 332, 335, 535
~-Software-Ingenieur 453–456
~-Software-Plattformen 221
~-Software-Projekt 78
~-Softwarebasis 282
~-Softwareproduktion 100
~-Thin-Client 284
~-Ticket-Request-System .. *siehe* OTRS
~-Verfahren 76
~-Verfahrensweise 108
~-Zertifikat 365
Achillesferse von ~ 109
Bugs pro Codezeile in ~ 148
Entwicklung von ~ 148
proprietäre ~ 150
Verlässlichkeit von ~ 109
OpenCola 104
OpenCoop 114
OpenCourseWare 523, 526–528
OpenDocument 242, 387
OpenLDAP 241, 249, 535
OpenOffice.org . 223–237, 242, 370, 387, 535
OpenSchoolServer 368
OpenUsability 289, 291–295
OpenVZ 265–268
Oracle 48, 214
Orth, Helen 345
Osborn (Missouri) 118
OScar-Projekt 98
OSCON-Konferenz 371
OSI 535
OSS 535
Österreich 105
OTRS 244

P

PageRank 178
Papierpost 240
Paradigmenwechsel 453–456
Pashtu 380
Patch 535

Patcher (Software) 333
Patent 74, 79, 81, 95, 230, 512
~recht 62
~ Gratis lizensieren 105
~ auf Bakterium 105
~-schutz 512
~anspruch 230, 512, 519
~ierbarkeit 230, 513
~ierung 170
~inhaber 512
~lizenz 512
~pool 516
~recherche 516
~rechtpools 521
~verletzung 230, 513
~wesen 512
Erzeugnis~ 514
Software~ 105, 230, 459, 476, 511–521
Verfahrens~ 513
Verzicht auf ~ 105
Pay-per-View-Verfahren 433
pd 333
Peace Corps 381
Peer production 91, 92
commons-based ~ 93, 96
Peer-Review ... 31–34, 39, 41, 42, 46, 50, 535
Pentium-I-PC 384
Perens, Bruce 47, 69
Peripherie 264
Perl 18, 28, 535
Permakultur 108
Persona 290
Personaldaten 245
Petersburger Beschluss 378
petitio principi 465
Pharmaindustrie 74, 75, 77, 79, 81, 84
Phasenübergang 409
Photoshop 387
PHP 28, 274, 535
Piazza Telematicha 111
pkgsrc 325
Plachetka, Uwe 119
Planwirtschaft 159, 168
Platform Configuration Register 303
Plug-in 535
policy 311
Portfolio-Analyse 201
POSIX 316
PostgreSQL 247
Praktiken und Verfahren der Partizipation 481
Preis 228
~differenzierung 429
~system 169
Preprint 398
Pressekodex 490
Printing on Demand 405, 408, 411
Privatkopie 428
Prix Ars Electronica 523
Processing 336
Procter & Gamble 93
Productivity Tools 130
Produkt
~diversifizierung 429
~haftpflichtrecht ... 502, 504, 506, 507
~haftrecht 508
~haftungsrecht 499, 500, 507
~management 88
Billig~ 106
Produktion 106
~smittel 107, 115
~soptimierung 168
~sprozess 106
Eigen-~ 105
kundennahe ~ 115
niederskalare ~ 115
Organisation der ~ 108
Produzentennetzwerk 112, 115
Programmierer .. 66, 211, 212, 214, 218, 221, 222, 496
Programmiersprache 26
Prohibitivpreis 187
Project Management Committee 318
Projektorganisation 488
kooperative ~ 488, 491
proprietär 225–231, 535
proprietäre Produktionsform 106
Prosuming 107
Proxy 535
Prozess
~modell 487
evolutionärer ~ 481, 483
strukturierter ~ 488
Verkauf von ~en 160
Public Domain 472
Publikation 415
~skrise 420
~sverzeichnis 398, 399
~swege 422
Puckette, Miller 333
PunkBuster 63
Pure Data ..... *siehe* pd
PXES 281
Python 18, 28, 535

Q
QPL ... 266, 535
Qualitätssicherung ... 92

R
Radiolehrstück ... 484
Raimi, Sam ... 337
Rammert, Werner ... 484
Raqs Media Collective ... 330
Rational-Choice-Theorie ... 62
Raymond, Eric S. ... 22, 47, 48, 66, 132, 487
RDP ... 278
Reach-through-Bestimmung ... 83
reacTable ... 334
Reas, Casey ... 336
Recht
~ am Notenbild ... 444
~edurchsetzung ... 444
~einhaber ... 427, 429
~sübertragung ... 414
~sunsicherheit ... 228
Eigentums~ ... 75
graphisches ~ ... 444
Patent~ ... 21, 36, 81
Straf~ ... 496
Urheber~ ... 21, 36, 79
Urhebernutzungs~ ... 444
Vergabe~ ... 227, 236
Red Hat ... 48, 130, 215, 499
~ Enterprise Linux (RHEL) ... 269
Redefreiheit ... 421
Redmond ... 228
Reed, David P. ... 485
Regelkreise ... 107
Regeneratives
~ Wissen ... 116
~ Design ... 117
Region ... 111
Regionalentwicklung ... 111
Registratur ... 240
Release ... 535
Rentabilitätsschwelle ... 83
Replikator ... 153
Repository ... 398–403, 420
Central ~ ... 398, 399, 401, 405
Institutional ~ ... 398–400, 404, 405, 408
Personal ~ ... 398, 399
Reproduktionsrate ... 108
Reproduktionstechnologie ... 104
Request-for-Comments-Verfahren ... 33
Ressource Management Layer ... 307
Ressourcen ... 105
Erschöpfung von ~ ... 104
regenerative ~ ... 117
Retrieval ... 403
Return On Investment ... *siehe* ROI
Revolution
Denk~ ... 108
Reziprozität ... 63
Rheingold, Howard ... 114, 483
Ritchie, Dennis ... 27
ROI ... 253
Roosendaal, Ton ... 337
Rootkit ... 322
Roundtrip ... 277
Rundfunk ... 483

S
Saatgut ... 105
SAGA-Dokumente ... 242
Saint-Exupéry, Antoine de ... 482
Salzer, Jerome H. ... 485
Samba ... 241, 249, 535
Samll, Craig ... 145
Sandvig, Christian ... 486
SANE ... 359
SAP ... 225, 291
SAS-II ... 370
Schadensersatz ... 514
Schlüsselqualifikationen ... 344
IT-~ ... 343
Schneeballeffekt ... 530
Schnittstelle ... 234, 236
Benutzer~ ... 213, 218
offene ~ ... 213, 221, 234
Programmier~ ... 213, 216
proprietäre ~ ... 213, 216, 221
Schulserver ... 356
Schulung ... 229–233, 344
Schulwesen ... 366
Schwäbisch Hall ... 274, 281, 285, 287
Schwarmverhalten ... 174
Scientific management ... 89
Sealing ... 305
Second Life ... 116
Secure boot ... 304
Secure GUI ... 311
Secure shell ... *siehe* SSH
Security Advisory ... 322
Security Kernel ... 307
Security Policy ... 305
Security Token ... 244, 287
Selbstmotivation ... 92
Selbstorganisation ... 91, 92, 174

Selbstselektion 91, 92
Seminarix 359
Sendmail 18, 19, 28, 535
Server 226–236, 536
server-based computing 273–288
Serverbetriebssystem 206
Seven Mile Boots 331
Sharecropping 466
sharing 461
Sicherheit 202, 217, 219, 220, 222
~splattform 301
Silvi, Guiseppe 111
SIM 287
SINA 232, 241
~ Native Client 244
~ Thin Client 244
Singapur 240
single point of administration 285
Sketchup 116
Skolelinux 362
Skript 536
Smartcard 287, 536
SMB/CIFS 387
SMTP 31
Software 24, 61, 62
~ als Ware 354
~-Installation 323
~-Stack 282
~bündelung 126, 130
~entwickler 211, 213
~entwicklung 212, 221, 502
~industrie 210, 211, 221, 222
~lizenz 511
~markt 210, 211
~qualität 217
~überlassung 498
Application-Service-Providing-~ .. 204
Cluster~ 200, 204
Co-Design-~ 113
freie ~ ... 23–25, 27–30, 35, 41, 46–48, 132, 196, 203, 204, 207, 224–238
GNU-~ 214
Grafik~ 198, 200, 202
Grid-~ 204
kommerzielle ~ 196
Open-Source-~ 87, 95, 201, 205–207, 210, 211, 216, 221, 496–504, 506–508
Open-Source-Design-~ 97
proprietäre ~ 29, 67, 206, 210–212, 214, 216, 220, 221, 230, 503, 504, 507
Social ~ 375
Stata-~ 96
Struktur der ~industrie 133
Time-Sharing-~ 26
Unternehmen differenzierende ~ .. 138
Video~ 200, 201, 204
wissenschaftlich-technische ~ 198, 200–202
Softwareentwicklung
~ in Open-Source-Communitys ... 483
Anforderungen an ~sprozess 487
ko-evolutionärer ~sprozess 483
Methoden der ~ 139
Solar Power Village 118
Solar-Roof-Technologie 108
Solaris 214
Soleri, Paole 108
Sondheim, Alan 332
Sonnenenergie 108
Sozialbezüge 481
Soziale Normen 485
soziokultureller Ursprung 20
SpaceWar 25, 26, 67
SPAM 299
SpamAssassin 255
SPARC Author's Addendum 422
Sphinx 368
Spindler-Gutachten 496
Sprachapplikation 198, 200
Sprachverarbeitung 197, 205
Springer 403, 409, 411
Spyware 63
SQL 536
Squid 241
SSH 278
SSL 536
Stabilitätspakt Afghanistan 378
Stallman, Richard .. 22, 29, 34, 35, 39, 40, 43, 47, 69, 132, 354, 533
Standard
ISO-~ 237
offener ~ 233–237
StarOffice 370
Stata Corp. 96, 99
Steffen, Gabriele 111
Stephenson, Neal 3
Steuer
~anreiz 159
~zahler 157, 232
Stigmergie 174–176
Direkte ~ 175
Indirekte ~ 176
Stoffdurchsatz 108

Stoffkreisläufe 117
Stone, Allucquere Rosanne 25
Storage Area Networks 257
Storage Root Key 303
Studie 228–236
  Machbarkeits~ 235, 236
Suber, Peter 406
Substitutionsstrategie 368
Subventionsprogramme 159
Subversion repository 246
Südafrika 111
Sun Microsystems 46, 214, 230, 535
Super Mario Clouds 330
Support 201, 202, 285
  ~verträge 243
SUSE 48, 226, 230, 283, 372, 382
  ~ Open Enterprise Server 372
Symbiosesystem 108
Symbolketten 103
Synthetische Biologie 116
System 224, 229, 231, 236
  komplexes ~ 107
  Meta-~ 107
  regeneratives ~ 108
  Symbiose-~ 108
Systems-Management 285–286, 288
systrace 323

T

Tantiemen 444
Tauschkreis 114
Taylor, Frederick 44, 89
Taylorismus 20–22, 25, 44, 51, 52
TCG 302
TCP 278, 387
TCP/IP 18, 31
Technik
  ~entwicklung 485
  ~vergessenheit 485
  digitale ~ 25, 51
Technische Schutzmechanismen .. *siehe* TSM
Technologie
  ~-Austausch 110
  abfallfreie ~ 117
  digitale ~ 20–22, 26, 52
  industrielle ~ 20
  personal ~ 115
  Solar-Roof- ~ 108
Technology Preview 257
Telecenter 112
TEMPEST 334
Temporary Autonomous Sarai 330
Terminalserver 227, 235
  Dimensionierung 284
  Linux ~ 273–288
Terroristenbekämpfung 468
TFTP 277, 280
  Multicast ~ 277
Thin Station 281
Thin-Client 273–288, 536
  ~-Hardware 283
  ~-Hersteller 284–286
  ~-Konzept 276
  Diskless-~ 279–281
  heterogene ~-Umgebung 280
  Open-Source-~ 284
  Systems-Management für ~s . 285–286
thinking center 476
Thompson, Ken 27
Thread 536
Threadless 88–99
Thunderbird 224, 242
Time Sharing 24, 474, 476
Todd, John 107, 117
Toffler, Alvin 107
Toll Access 396, 397, 402
Torvalds, Linus . 43, 45, 46, 51, 134, 175, 335
Total Cost of Ownership . 191, 205–207, 215, 536
TPM 303
Trans Bacter (Bakterienstamm) 105
Transaktionskosten 169, 483
Transmeta 281
Trinux 219
Trittbrettfahrer 161, 162, 169
Trivial File Transfer Protocol .... *siehe* TFTP
Trojanisches Pferd 300
Trolltech 151
Tropical Disease Initiative 77, 79
Trusted
  ~ Computing 302
  ~ Computing Group ..... *siehe* TCG
  ~ Software Layer 307
  ~ bootloader 304
  ~-Boot-Vorgang 304
  ~-Platform-Modul ..... *siehe* TPM
TSM 427, 429, 431
Turaya 302, 307
  ~ Pilot 311
Tuschek, Elisabeth 344

U

Ubuntu 282–284, 288
UHD 246

Umbrello, UML-Software 348
Umgehungsschutz 429
UML 347, 348
Unabhängigkeit
  Hersteller~ 235–237
  Plattform~ 231
UNESCO 409
United Nations Development Programme 386
Univention 285
  ~ Corporate Server 285
Universitäre Lehre 343
Universität
  ~ Bremen 343
  ~ Minho 400–402
  ~ Southampton 400, 402, 406
  ~sbibliothek 405–407
University of California 157
UNIX 27–29, 43
unsichtbare Hand 169
Unterricht
  ~scurriculum 376
  ~sscript 355
  Frontal-~ 355
  Projekt-~ 355
Update 227–236
Urheber 427
Urheberrecht 62, 70, 302, 407, 414
  ~sgesetz 433
  ~sinhaber 415
  ~sreform 433
  ~sschutz 528
  ~sverfügung 397
Usability 289–298, 536
  ~-Studienmöglichkeiten 290
  ~ guidelines 296
  ~ und Open-Source-Projekte . 291–297
  ~-Community 291
  ~-Experte 290
  IBM und ~ 291
  Microsoft und ~ 291
  SAP und ~ 291
USB over IP ..... *siehe* USB/IP
USB-Token 287
USB/IP 287, 288
USB/SIM 288
USB/SIM-Authentisierung 287
Usenet 28, 536
User Help Desk 246

V

van Rossum, Guido 28
Verbraucherschutz 499
Verfeinerte Auffassung der Natur 107
veriexec 322
Verlag 397, 422
  Noten-~ 444, 446
Vernetzung 18, 43
  digitale ~ 27, 31, 33, 34, 42, 50, 53
  globale ~ 17, 28, 52
Versammlungsfreiheit 529
Vertrag
  ~sschluss 423
  ~svorlage 415
  Garantie~ 519
  Paket~ 410
Verwaltung
  Bundes~ 233, 242
  öffentliche ~ 223–238
  Senats~ 228, 234–235
  Stadt~ 227, 230
Verwertungsgesellschaft .. 109, 418, 419, 423
Verwertungsrechte
  exklusive ~ 487
Verzeichnisdienst 226, 245, 536
VIA C3 281
Video
  ~ Bridge 112
  ~applikation 198
  ~plattform 529
  ~software 197
Vintage-Faktoren 344
Virtual Private Network 241
Virtualisierung 226, 263–271, 273, 308
  ~sprodukt 264
  ~sschicht 264, 269
  Betriebssystem~ 265
  Hardware~ 264
  Komplett~ 264
  Para~ 264, 269
virtualization overhead 264
Virtuelle
  ~ Gemeinschaft 33, 68
  ~ Kollaboration 43, 44
  ~ Objekte 116
Virtuosität 375
Virtuozzo 265
VMM (Virtual Machine Monitor) 265
VMS 536
VMware 226, 264
VNC 278
Volkshochschule 527
von Hippel, Eric 293
von Neumann, John 25

VPN 241, 283

W

Wall, Larry 28
Waren
~-Konsumption 106
~-Produktion 106
Wartung 285
Wartungskosten 243
webbasiert 231, 242
WebCT 374
Weblog ..... *siehe* Blog
Webmin 241
WebOpenSchool 358
Webservice 229, 488
Webspace 529
WebSphere 214
Web 2.0 ..... 489, 490
Wellcome Trust 400, 402, 410
Wells, H. G. 37
Werkart 441, 449
Werkstofftechnologie 117
Werkverzeichnis 443
Wertschöpfung 90, 93, 96
~skette 90, 395, 404, 407, 408
~snetzwerk 91
arbeitsteilige ~ 92, 98
industrielle ~ 96
interaktive ~ 88–100
Wettbewerb 129, 168, 185, 213, 231–237
Wiederverwendung von Quellcode 189
Wien 365
Wiener Erklärung 431
WIENUX 525
Wiki 417, 536
Wikipedia 53, 92, 166, 374, 417, 418, 528, 529
Windows .. 130, 206, 214, 216–219, 224–238
~server 241
Windows Vista 228–236, 312
Belastungen des Bildungsetats 352
Datenübermittlungen durch ~ 352
Nutzungsbedingungen von ~ 352
Validierungszwang von ~ 352
Wine 226, 387
Wirtschaft
~ eines kapitalistischen Staates 159
~sleben 21
Wirtschaftlichkeit 205–207, 235, 237
Wirtschaftszweige 110
Wissen
~saustausch 527
~sdiffusion 188, 190, 191
~stransfer 74
~sverknappung 407, 408
digitale ~sallmende 526
freie ~skultur 362
geteiltes ~ 353
Offenes ~ 113
regeneratives ~ 116
vernetzte ~sarbeit 22
Wissenschaft 420
~liche Werke 421
~sfreiheit 421
Aufblühen der ~ 108
WLAN 525
Wohlstandseffekte 109
Wohltätigkeit ..... *siehe* Altruismus
Wolfram, Stephen 107
Wollmux 224, 229
Workflowsystem 245
World Wide Web 41–44, 49, 166, 387
Wurm 322

X

X Window System 274–277
X-Client 275–277
x-manage 242
X-Protokoll 275–278, 282
Opcode 277
X-Server 275–277
X.25 240
X11 ..... *siehe* X Window System
Xandros 283
XDMCP 282
Xen 268–271
XML 234, 536

Z

Zahlungsbereitschaft 91, 187
Zeitgenössische Musik 445
Zeitschriftenmarkt 428
Zeitzonen 240
Zelluläre Automaten 107
Zentralkommitee 176
ZiiK 378
Zitierfreiheit 417
Zuboff, Shoshanna 113
Zufällige Mutation 107
Zugangsgebühren 430
Zugangssperren 429
Zukunftstechnologien 274
Zwei-Faktor-Authentifizierung 301

Mitwirkende

Daniel Auener studiert Informatik an der Technischen Universität Berlin. Sein Studienschwerpunkt ist „Informatik und Gesellschaft". Neben der Akquise und Betreuung von Autoren hat er die Webseite des Open Source Jahrbuches restrukturiert und vollständig technisch überarbeitet.

Matthias Bärwolff ist wissenschaftlicher Mitarbeiter im Fachgebiet Informatik und Gesellschaft der Technischen Universität Berlin. Als Mitherausgeber des Buches hat er Anmerkungen und Überarbeitungen für sämtliche Texte im Buch eingebracht und damit maßgeblich die Qualitätssicherung des Buches übernommen.

Jürgen Bitzer (Dr.) ist wissenschaftlicher Assistent an der Freien Universität Berlin. In seiner Forschung beschäftigt er sich mit Wissensspillovereffekten, Wirtschaftswachstum und Open–Source-Software. Seine Forschungsergebnisse wurden regelmäßig in internationalen Zeitschriften veröffentlicht, z. B. in *Economics Letters*, *Journal of Comparative Economics*, *Applied Economics*, *Kyklos*, *Information Economics and Policy* und in *Economic Systems*.

Lina Böcker (Dipl. iur.) ist wissenschaftliche Mitarbeiterin am Institut für deutsches und europäisches Wirtschafts-, Wettbewerbs- und Energierecht der Freien Universität Berlin. Sie promoviert über das Thema „Rechtliche Aspekte bei Freier, insbesondere GPL-Software" am Lehrstuhl von Prof. Dr. Dr. Dr. Franz Jürgen Säcker. Neben dem Recht des geistigen Eigentums liegen ihre Schwerpunkte im Kartell- und Telekommunikationsrecht.

Nicole Bräuer hat das Lektorat des Buches durchgeführt, was sie sehr flexibel und zu unserer vollsten Zufriedenheit erledigt hat.

Carsten Brunke ist Gründer und Geschäftsführer der *inmedias.it GmbH*. 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.

Maika Büschenfeldt ist Diplom-Soziologin und studierte an der Universität Hamburg neben ihrem Hauptfach auch Informatik und Psychologie. Sie ist seit Januar 2002 als wissenschaftliche Mitarbeiterin am FB Mathematik/Informatik der Universität Bremen (AG „Digitale Medien in der Bildung") beschäftigt.

Angela Dirlick ist Übersetzerstudentin (Englisch/ Französisch) an der Humboldt-Universität zu Berlin. Sie arbeitete an der Übersetzung des Artikels von Bruce Perens mit.

Leonhard Dobusch ist studierter Jurist und Betriebswirt (Universität Linz) und promoviert im Graduiertenkolleg „Pfade organisatorischer Prozesse" am Fachbereich Wirtschaftswissenschaften der Freien Universität Berlin. Das Thema seiner Dissertation lautet „Wege aus sozio-technologischer Pfadabhängigkeit: GNU/Linux-Migration in großen Stadtverwaltungen".

Christina Dörge ist Diplom-Informatikerin und arbeitet als wissenschaftliche Mitarbeiterin in der Arbeitsgruppe „Digitale Medien in der Bildung (DiMeB)", Fachbereich Informatik, Universität Bremen, wo sie am Aufbau des „Zentrums für Interaktion mit Digitalen Medien (ZIM)"

beteiligt ist. Zu ihren Hauptaufgaben gehören weiterhin die Entwicklung von Lehrveranstaltungen im Bereich „IT-Schlüsselqualifikation".

Christian Dorninger (Dr.) ist Abteilungsleiter für Schulentwicklung und IT-Angelegenheiten im Österreichischen Bundesministerium für Bildung, Wissenschaft und Kultur. Als Mitglied der IT-Lenkungsgruppe ist er für Softwarebeschaffung und Projektbegleitung von IT- und E-Learning-Projekten im Schulbereich verantwortlich. Nebenberuflich ist er zudem Lehrender an einer Wiener HTL und Mentor von Kursen der Fernuniversität Hagen, in den Bereichen „angewandte Informatik" und „Mathematik".

Thomas Dreier (Prof. Dr. iur.) ist Leiter des Instituts für Informationsrecht an der Universität Karlsruhe. Zuvor war er am Max-Planck-Institut für geistiges Eigentum in München tätig. Prof. Dreier ist Vizepräsident der *Association littéraire et artistique internationale (ALAI)* und Geschäftsführer der *Deutschen Gesellschaft für Recht und Informatik e. V. (DGRI)*. Seine Forschungsschwerpunkte sind Urheberrecht und neue Technologien sowie Rechtsfragen der Informationsgesellschaft. Unter seiner Leitung wurde die deutsche Fassung der Creative–Commons-Lizenzen erarbeitet.

Elke Eisenschmidt studiert Übersetzen auf Diplom an der Humboldt-Universität zu Berlin im 7. Semester, mit den Sprachen Englisch, Deutsch und Norwegisch. Als Vertiefungsfach hat sie Bauingenieurwesen belegt. Für dieses Buch hat sie den Text „Pharmaforschung mit Open-Source-Methoden" von Janet Hope übersetzt.

Hubert Feyrer hat an der FH Regensburg Informatik studiert und sich dort auf Betriebssysteme und systemnahe Anwendungen spezialisiert. Neben dem Abhalten von Vorlesungen zu den Themen „Systemadministration" und „Open Source" arbeitet er am NetBSD-Projekt mit, wo er die Infrastruktur für Anwendungen von Drittanbietern mit aufgebaut hat sowie Öffentlichkeitsarbeit betreibt. Er arbeitet momentan an seiner Dissertation über ein Übungssystem für den Bereich „Systemadministration" während er Vorlesungen über Systemadministration und Open Source gibt.

Fabian Flägel ist Student der Informatik an der Technischen Universität Berlin. Sein Studienschwerpunkt ist „Informatik und Gesellschaft". Er unterstützte das Team mit seinem Wissen aus dem Open-Source-Jahrbuch-Projekt 2006 und setzte zusammen mit Matthias Liebig ein automatisches Buildsystem für das Buch auf.

Gundolf S. Freyermuth (Prof. Dr.) lehrt Angewandte Medienwissenschaften an der Internationalen Filmschule Köln. Seine Lehr- und Forschungsschwerpunkte betreffen die kulturellen Konsequenzen der Digitalisierung. Vor seiner Lehrtätigkeit arbeitete Freyermuth als Redakteur, Reporter und freier Autor. Er hat 13 Romane und Sachbücher sowie rund 400 Artikel veröffentlicht. Gegenwärtig schließt er eine Kulturgeschichte der Digitalisierung ab.

Robert A. Gehring studierte Elektrotechnik, Informatik und Philosophie an der Technischen Hochschule Ilmenau und an der Technischen Universität Berlin. Dort promoviert er derzeit zu Fragen der Softwareökonomie und des Softwarerechts. Seit Ende 2005 ist er Mitglied der Redaktion des Urheberrechtsportals *iRights.info*, das 2006 mit dem Grimme-Online-Preis ausgezeichnet wurde. Als freiberuflicher Consultant berät er namhafte deutsche Unternehmen in IT-Fragen. Neben der Wahrnehmung der Aufgaben als Herausgeber hat Robert Gehring für dieses Jahrbuch den Artikel von Richard Stallman übersetzt.

Martin Görnitz ist Student der Informatik an der Technischen Universität Berlin. Die Schwerpunkte seines Studiums bilden „Wirtschafts- und verwaltungsorientierte Anwendungen der Informatik“ und „Informatik und Gesellschaft“. Für das Jahrbuch hat er das Kapitel „Im Dialog: Recht, Politik und Open Source“ betreut und die Übersetzung des Beitrags von Bruce Perens überarbeitet.

Nico Gulden studierte über die *IBM Deutschland* Angewandte Informatik an der Berufsakademie Stuttgart. Während seiner Assistentenarbeit im Entwicklungslabor der *IBM* in Böblingen erarbeitete er im Team eine Methode für die Verifikation von *floating-point units*, die zum Patent angemeldet wurde. Über verschiedene Praxiseinsätze im Entwicklungslabor der *IBM* arbeitete er in der Systemadministration und der Hard- und Softwareentwicklung. Derzeit verantwortet er die technische Projektarbeit in der *Linux Solutions Group e. V.*

Hans Henning studiert an der Technischen Universität Berlin Informatik; seine Studienschwerpunkte sind „Softwaretechnik“ und „Informatik und Gesellschaft“. Neben der Betreuung von Autoren und dem Setzen von Artikeln organisiert er den Auftritt des Jahrbuches auf der nächsten „Langen Nacht der Wissenschaften“.

Francis Heylighen ist Professor am „Center Leo Apostel“ der *Vrije Universiteit Brussel* und leitet dort die Forschungsgruppe *ECCO*. Er promovierte 1987 in Physik. Zurzeit arbeitet er über Selbstorganisation und Evolution von komplexen kognitiven Systemen. Heylighen ist Herausgeber des Principia-Cybernetica-Projekts, dessen Website, eine der ersten komplexen und interaktiven Sites überhaupt, er 1993 initiierte.

Eberhard R. Hilf (Prof. Dr.) ist CEO der *Institute for Science Networking Oldenburg GmbH* an der Carl von Ossietzky Universität, die Open-Access-Mehrwertdienste plant und entwickelt, sowie der *Scineo* die im Verbund mit verwandten Firmen Bibliotheken und Verlage technisch und bei der Umgestaltung ihres Geschäftsmodells und des Einsatzes neuer Dienste berät. Er ist Mitherausgeber des *Journal of Small Systems*.

Birgit Hollenbach ist Diplom-Biochemikerin mit Studium an der FU Berlin und viel Berufserfahrung. Sie ist angehende Diplom-Übersetzerin und Dolmetscherin für die Sprachen Englisch und Spanisch an der HU Berlin. Sie übersetzte den Artikel von Francis Heylighen aus dem Englischen ins Deutsche.

Janet Hope ist Biochemikerin und Molekularbiologin und praktizierte als Anwältin. Sie hat ihre Doktorarbeit über „Open Source Biotechnology“ an der *Australian National University* im Jahr 2004 abgeschlossen. Ihr erstes Buch „Bio Bazaar: Biotechnology and the Open Source Revolution“ wird voraussichtlich 2007 bei *Harvard University Press* erscheinen.

Christoph Ihl ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Betriebswirtschaftslehre – Information, Organisation und Management an der Technischen Universität München. Im Rahmen seiner Forschungstätigkeit beschäftigt er sich mit empirischen Analysen des Konsumentenverhaltens bei technologiebasierten Leistungsangeboten.

Kei Ishii ist der Sohn des Komponisten Maki Ishii. Er studierte an der Technischen Universität Berlin Informatik und war dort wissenschaftlicher Mitarbeiter am Fachgebiet Informatik und Gesellschaft (Prof. Bernd Lutterbeck). Er hat kürzlich über das Thema „Regulierung durch Technologie“ („Code Governance“) promoviert.

Holger Kett ist seit über sechs Jahren als wissenschaftlicher Mitarbeiter und Projektleiter am Fraunhofer IAO tätig. Seine Themenschwerpunkte sind die elektronische Beschaffung, Marketing und Vertrieb mit Fokus auf Unternehmen und öffentliche Verwaltungen. Er ist Projektleiter im Fraunhofer Electronic-Business-Innovationszentrum. Er hat im Rahmen seiner Tätigkeiten neben Forschungsprojekten auch zahlreiche Beratungsprojekte in Unternehmen begleitet und durchgeführt.

Alexander Knorr (Dr.) ist wissenschaftlicher Assistent am Institut für Ethnologie und Afrikanistik der Ludwig-Maximilians-Universität München. Seine Forschungsinteressen liegen im emergenten ethnologischen Teilgebiet Cyberanthropology. Sein Habilitationsprojekt „max-mod::online among the gamemodders – the cultural appropriation of information and communication technologies" kann online auf seinem Weblog http://xirdal.lmu.de/cgi-bin/blosxom.cgi verfolgt werden.

Stefan Krempl (Dr.) arbeitet als freier Autor in Berlin. 2005 promovierte er zum Thema „Medien, Internet, Krieg: Das Beispiel Kosovo". Er ist Autor diverser Bücher und publiziert regelmäßig in Magazinen und Online-Diensten wie c't, heise online und Telepolis sowie in Tages- und Wochenzeitungen von der Süddeutschen Zeitung über die Neue Zürcher Zeitung bis zur Zeit schwerpunktmäßig über politische, rechtliche sowie kulturelle Themen rund um Internet und Informationstechnik.

Christian Laux (Dr.) ist Rechtsanwalt und studierte Rechtswissenschaften in Zürich und Paris. Dissertation zu den Grundlagen der Auslegung von Verträgen über urheberrechtlich geschützte Inhalte („Vertragsauslegung im Urheberrecht", Bern 2003). LLM-Studium an der *Stanford Law School* (2006–2007). Christian Laux arbeitet als Rechtsanwalt in einer Wirtschaftskanzlei in Winterthur. Mitbegründer des Vereins „Openlaw – Plattform für Recht und freie Software".

Matthias Liebig studiert Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Offene Kommunikationssysteme" und „Informatik und Gesellschaft". Er kümmerte sich hauptsächlich um Textsatz sowie Layout und entwarf ein automatisches Buildsystem für die Jahrbuch-Quellen. Ohne ihn würde es dieses Buch nicht geben.

Markus Linnemann ist wissenschaftlicher Mitarbeiter im Bereich *Trusted Computing* des Instituts für Internet-Sicherheit an der FH-Gelsenkirchen.

Bernd Lutterbeck (Prof. Dr. iur.) studierte Rechtswissenschaften und Betriebswirtschaftslehre in Kiel und Tübingen. Seit 1984 ist Bernd Lutterbeck Professor für Wirtschaftsinformatik an der Technischen Universität Berlin mit den Schwerpunkten Informatik und Gesellschaft, Datenschutz- und Informationsrecht sowie Verwaltungsinformatik. Seine Arbeitsschwerpunkte sind E-Government, Theorie und Praxis geistigen Eigentums, Open Source und *European Governance*.

Reto Mantz (Ref. iur.) studierte Rechtswissenschaften in Göttingen und Frankfurt und studiert derzeit Informatik an der Universität Frankfurt. Nach einer Tätigkeit als wissenschaftlicher Assistent am Lehrstuhl von Prof. Dr. Gerald Spindler, Universität Göttingen, ist er zur Zeit wissenschaftlicher Mitarbeiter der Anwaltssozietät *Heymann & Partner*, Frankfurt. Er promoviert über das Thema „Offene Netzwerke".

Joscha May studiert Wirtschaftsingenieurwesen an der Technischen Universität Berlin. Seine Studienschwerpunkte sind Informations- und Kommunikationstechnologien sowie Innovati-

onsmanagement und -marketing. Er ist der diesjährige Gewinner des Open-Fiction-Schreibwettbewerbs.

Katharina Mertens ist IFK-Studentin an der Humboldt-Universität zu Berlin und hat an der Übersetzung des Textes von Bruce Perens mitgewirkt.

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. Nach zwei Jahren als Systemanalytiker wurde er dort Leiter der Sparte Netzwerke und Datenbanken. Seit 2004 ist als Part einer Doppelspitze für die IT-Services der Norddeutschen Affinerie AG verantwortlich, hier insbesondere für den Bereich IT-Infrastruktur und Telekommunikation.

Franz Nahrada ist von der Ausbildung her Soziologe und Philosoph, dem die gesellschaftsverändernde Rolle von Wissenschaft ein Anliegen war und ist. Nach weiteren Lehrjahren in der Computerbranche und vielen Studienreisen vor allem in die USA hat er sich in den letzten 17 Jahren vor allem der Frage der Kombination dörflicher, naturnaher Lebensräume mit den Möglichkeiten der modernen Kommunikationstechnologie verschrieben. Er war Mitveranstalter der Wiener Oekonux-Konferenz.

Andreas Niederländer ist Geschäftsführer der *iKu Systems & Services* und beschäftigt sich seit 1995 mit dem Einsatz von Open-Source-Software für Unternehmen.

Bruce Perens ist einer der Begründer der Open-Source-Software-Bewegung und Verfasser der Open-Source-Definition. Er ist Herausgeber der mittlerweile 13 Bände umfassenden *Bruce Perens' Open Source Series*. Perens betreibt eine Firma für strategische Beratung, *Perens LCC*, die sich auf Linux und Open Source spezialisiert hat. Er ist *Senior Research Scientist* für Open Source am *Cyber Security Policy Research Institute* der *George Washington University*.

Nazir Peroz (Dr.) geboren in Kabul/Afghanistan, studierte Informatik an der TU Berlin, wo er 1992 promovierte. Er ist Leiter des Zentrums für internationale interkulturelle Kommunikation der TU Berlin sowie des Arbeitsgebiets „Informatik und Entwicklungsländer" an der TU Berlin. Seit 2002 engagiert er sich am Wiederaufbau der akademischen Strukturen in Afghanistan und ist seit 2004 IT-Koordinator Afghanistans.

Martin Pichlmair (Dr.) ist in Wien lebender Medienkünstler und Wissenschaftler. Er ist als Universitätsassistent am Institut für Gestaltungs- und Wirkungsforschung der Technischen Universität Wien tätig. Seine Arbeiten wurden in zahlreichen Ausstellungen wie der *ISEA*, dem *Ars Electronica Festival* und dem *Microwave International Festival for New Media Art* gezeigt. In der Forschung beschäftigt er sich mit der Theorie und Praxis von interaktiver Kunst und Design.

Frank Piller (Dr.) ist Privatdozent für Betriebswirtschaftslehre an der TU München und forscht derzeit an der *MIT Sloan School of Management*, Cambridge, USA. Seine Forschungsinteressen liegen im Bereich Information und Kommunikation, Technologie- und Innovationsmanagement und kundenzentrierter Wertschöpfungsstrategien (www.open-innovation.com).

Norbert Pohlmann (Prof. Dr.) ist Informatikprofessor für verteilte Systeme und Informationssicherheit sowie Leiter des Instituts für Internetsicherheit an der FH-Gelsenkirchen.

Herbert Reckmann (Dr.) ist Vorstandssprecher des in Deutschland bundesweit agierenden gemeinnützigen Vereins *Freie Software und Bildung (FSuB e. V.)*. Vor seinem ehrenamtlichen

Engagement als Sprecher des *FSuB e. V.* war Dr. Herbert Reckmann als Erziehungswissenschaftler an der Universität Regensburg und der Fern-Universität Hagen in Forschung und Lehre tätig. Von 1985 bis 2002 arbeitete er am Landesinstitut für Schule (Soest) des Landes Nordrhein-Westfalen. Dort leitete er das Grundsatzreferat Lehrerfortbildung.

Ralf Reichwald (Prof. Dr. Prof. h.c. Dr. h.c.) lehrt Betriebswirtschaftslehre an der Fakultät für Wirtschaftswissenschaften und ist Inhaber des Lehrstuhls für Betriebswirtschaftslehre – Information, Organisation und Management (IOM) an der TU München. Seine Hauptarbeitsgebiete sind Führung und Organisation, Informations- und Kommunikationswirtschaft sowie Dienstleistungsmanagement.

Ellen Reitmayr ist freiberufliche Usability-Beraterin in Berlin. Als Mit-Gründerin und aktives Mitglied der Plattform www.openusability.org bringt sie Open-Source-Software und Usability zusammen.

Thomas Renner ist Leiter des „Competence Center Electronic Business" am Fraunhofer IAO. Im Fokus seiner Arbeiten stehen innovative Lösungen für zwischenbetriebliche Kooperationen und Geschäftsnetzwerke. Dazu gehören auch die Optimierung des Einsatzes von IT-Systemen und IT-gestützten Geschäftsprozessen sowie die Analyse von Geschäftsmodellen und die Entwicklung von E-Business-Strategien.

Sascha Rex (Dr.) studierte Wirtschaftswissenschaften an der Universität Augsburg. Von 2000 bis 2005 war er wissenschaftlicher Mitarbeiter am Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO). Tätigkeitsschwerpunkte waren XML-Technologien, Softwarearchitekturen für elektronische Geschäftssysteme und die Untersuchung von Einsatzmöglichkeiten freier Software in Unternehmen. Er ist inzwischen als freier Unternehmensberater tätig.

Christine Rolles ist Diplom-Übersetzerin für Französisch und Spanisch. Nach langjähriger Berufstätigkeit als europäische Projektkoordinatorin ist sie nun Studentin im Zweitstudium an der Humboldt-Universität, und zwar im Fachbereich „Interkulturelle Fachkommunikation". Für dieses Buch übersetzte sie den Text „Die Ökonomie der Softwaremärkte" von Hal Varian und Carl Shapiro.

Nadim Sarrouh ist Student der Informatik an der Technischen Universität Berlin. Seine Schwerpunkte sind „Systemanalyse und EDV" sowie „Informatik und Gesellschaft". Sein Aufgabenbereich lag neben der Betreuung von Autoren in der grafischen Gestaltung des Buches sowie der zugehörigen Werbung und Präsentation. Außerdem war er für die Aquise der Übersetzer in Zusammenarbeit mit Eva Affandi verantwortlich.

Gerhard Schöppe möchte seinen verstorbenen Lehrern und Freunden, dem Universalisten Merlin (Piet-Jürgen Fries), dem Aktionskünstler Der Kolter (Uwe Kolter) und dem Kabarettisten Wolfgang Neuss für ihre Lehren in Vielheit, Einklang und Ausdruck danken, die damit maßgeblich an seinem Teil zur Entstehung dieses Buches beigetragen haben. Schöppe studiert Informatik und Neurobiologie an der TU Berlin.

Philipp J. H. Schröder ist Professor für Volkswirtschaftslehre an der *Aarhus School of Business*, Dänemark. Sein Forschungsfeld umfasst die Bereiche Außenwirtschaft, Open-Source-Software, Transformation und Europäische Integration. Seine Forschungsergebnisse wurden regelmäßig in internationalen Zeitschriften veröffentlicht, z. B. in *Review of International Economics*, *Information Economics and Policy*, *Journal of Economics*, *Journal of Regulatory Economics*, *Canadian Journal of Economics* und dem *Journal of Comparative Economics*.

Nadja Schüler ist IFK-Studentin (Dolmetschen und Übersetzen/Englisch und Französisch) an der Humboldt-Universität zu Berlin und hat an der Übersetzung des Textes von Bruce Perens mitgewirkt.

Stefan Schumacher beschäftigt sich seit 1998 mit *UNIX* und seit 2001 mit *NetBSD* im Speziellen. Als Datenbankadministrator und -Entwickler befasst er sich ausgiebig mit *PostgreSQL*. Er unterstützt das NetBSD-Projekt mit Informations- und Werbematerial sowie Dokumentationen.

Carl Shapiro ist Transamerika-Professor für Unternehmensstrategie an der *Haas School of Business* und Chefberater bei *Charles River Associates*. Er beschäftigt sich seit über 20 Jahren mit der Ökonomie der Softwaremärkte und ist zusammen mit Hal Varian Autor des viel gelesenen Buches „Information Rules: A Strategic Guide to the Network Economy".

Richard Stallman gründete 1983 die Freie-Software-Bewegung und begann 1984 mit der Entwicklung des freien GNU-Betriebssystems. Ziel war und ist es, die Nutzung von Computern in freiheitlichen Gemeinschaften zu ermöglichen. In Verbindung mit dem Linux-Kernel hat sich das GNU-System zum heute führenden freien Betriebssystem entwickelt (GNU/Linux).

Sebastian Stein studiert an der Technischen Universität Berlin Informatik mit den Schwerpunkten „Kommunikationsnetze" und „Informatik und Gesellschaft". Er übernahm in diesem Jahr die Leitung des Projekts.

Kendra Stockmar ist Stipendiatin am Graduiertenkolleg IME (Information Management und Market Engineering) an der Universität Karlsruhe. Seit dem Studienabschluss an der RWTH Aachen beschäftigt sie sich am Institut für Informationsrecht der Universität Karlsruhe im Rahmen ihrer Forschungsarbeit mit dem Thema Open Access.

Jan Suhr ist Diplomand im Fach Informatik an der Technischen Universität Berlin und beschäftigt sich mit der Messung von Offenheit bei IT-Artefakten. Er war maßgeblich am Aufbau der Informatik-Fakultät an der Universität Herat, Afghanistan, beteiligt und repräsentierte zeitweise den Nationalen IT-Koordinator der höheren Bildung in Afghanistan. Darüber hinaus beriet Jan Suhr das Ministerium für Höhere Bildung in Afghanistan.

Daniel Tippmann schloss sein Studium der Soziologie an der FU Berlin mit der Abschlussarbeit „Gesellschaftsphänomen Open Source" ab. Derzeit ist er beschäftigt als wissenschaftlicher Koordinator am Zentrum für internationale interkulturelle Kommunikation (ZiiK) der TU Berlin. Seit 2005 arbeitet er am Aufbau der „Faculty of Computer Science" der Universität Herat in Afghanistan mit.

Silvia Varadinova ist Studentin der Informatik an der Technischen Universität Berlin. Ihre Studienschwerpunkte sind „Wirtschafts- und verwaltungsorientierte Anwendungen der Informatik" und „Theoretische Informatik". Neben der Betreuung von Autoren hatte sie Verantwortlichkeiten in der Organisation des Jahrbuchauftritts auf der CeBIT inne.

Hal R. Varian ist Professor an der *School of Information Management and Systems* und lehrt an der *Haas School of Business* und am Wirtschaftsinstitut der *University of California* in Berkeley. Er beschäftigt sich seit über 20 Jahren mit der Ökonomie der Softwaremärkte und ist zusammen mit Carl Shapiro Autor des Bestsellers „Information Rules: A Strategic Guide to the Network Economy".

Michael Vetter (Dr.) studierte Informatik mit Nebenfach Wirtschaftswissenschaften an der TU Darmstadt. Seit 2000 ist er Projektleiter im „Competence Center Electronic Business" am Fraunhofer Institut für Arbeitswirtschaft und Organisation (IAO). Seine Forschungs- und Arbeitsschwerpunkte sind Software-Architekturen, Software-Engineering und Internettechnologien für zwischenbetriebliche Geschäftsprozesse.

Mark Weinem wurde durch seine Beschäftigung mit dem Internet erstmals auf freie Software aufmerksam und setzt seit 1997 ausschließlich Open-Source-Lösungen ein. Als Anwender von BSD-Betriebssystemen unterstützt er seit 2005 *NetBSD*.

Annett Wendschuh studiert seit 2003 an der Humboldt-Universität zu Berlin Interkulturelle Fachkommunikation für Englisch und Portugiesisch und strebt einen Studienabschluss als Diplom-Übersetzerin und Diplom-Dolmetscherin für die genannten Sprachen an. Für dieses Buch übersetzte sie den Text „Einführung von Linux im öffentlichen Sektor" von Hal Varian und Carl Shapiro.

Torsten Werner hat an der Technischen Universität Dresden Physik studiert und ist seit dieser Zeit aktiver Debian-Entwickler. Danach war er mehrere Jahre als Assistent an derselben Universität mit den Aufgabenschwerpunkten Atom- und Plasmaphysik sowie Verkehrswirtschaft tätig. Seit drei Jahren arbeitet er in der Arbeitsgruppe „IT-Strategie" des Auswärtigen Amts.

Jan Widmer ist Rechtsanwalt und Mitglied von „Openlaw – Plattform für Recht und freie Software". Er studierte Rechtswissenschaft an der Universität Zürich. Derzeit studiert an der *Benjamin N. Cardozo School of Law*, New York (LLM-Studium mit Vertiefung Intellectual Property). Jan Widmer promoviert an der Universität Zürich.

Dennis Zimmer ist Consultant im Bereich Infrastruktur und Storage für das Unternehmen *Mightycare Solutions GmbH* sowie Zeitschriften- und Buchautor. Außerdem betreibt er die Community-Webseite www.vmachine.de, auf der Informationen zur Virtualisierung zu finden sind.

Bastian Zimmermann war Redakteur des Jahrbuchs 2006. Für dieses Buch übernahm er die Betreuung des Artikels von Stefan Krempl und wichtige Pressearbeiten im Hinblick auf die CeBIT.

Sponsoreninformationen

Im Folgenden listen wir sämtliche Sponsoren auf, die unser Projekt mit mehr als 1 000 Euro gefördert haben. Unsere Sponsorenrichtlinien finden Sie auf unserer Webseite unter http://www.opensourcejahrbuch.de/contact/sponsors. Die Sponsoren haben keinerlei inhaltlichen Einfluss auf dieses Jahrbuch oder kommende Jahrbücher.

Wir danken der *ZANOX.de AG* für ihre großzügige finanzielle Unterstützung in Höhe von 5 000 Euro.

Wir danken Professor Bernd Lutterbeck für seine großzügige Spende in Höhe von 3 500 Euro.

Die *ZANOX.de AG* ist Marktführer im erfolgsbasierten Multichannel-Commerce. Das Berliner Unternehmen bietet eine globale Lösung für die effiziente Produktvermarktung im Internet. Das Produktportfolio umfasst neben Affiliate Marketing und Suchmaschinen-Marketing auch E-Mail-Marketing, Online-Shopping sowie Customer-Loyalty-Programme. Aktuell setzen mehr als 1 500 internationale Top-Unternehmen auf den strategischen Vertriebskanal von *zanox*. Das Unternehmen wurde im Jahr 2000 gegründet und beschäftigt heute mehr als 260 Mitarbeiter.

zanox bietet seine Produkte im ASP-Modell an und nutzt inzwischen auf breiter Basis eine Java-EE-Systemarchitektur. So vertraut das Unternehmen beispielsweise auf *Subversion* und *Eclipse* in der Entwicklung sowie im Bereich Technologie/Libraries unter anderem auf *EJB3*, *Hibernate*, *JMS* und *Spring-MVC*. Auf Application-Server–Ebene setzt *zanox* heute auf *JBOSS*.

Lizenzen

Creative Commons

- Sie dürfen den Inhalt vervielfältigen, verbreiten und öffentlich aufführen.
- Nach Festlegung des Rechtsinhabers können die folgenden Bedingungen gelten:

  Namensnennung (BY) Sie müssen den Namen des Autors/Rechtsinhabers nennen.

  Keine kommerzielle Nutzung (NC) Dieser Inhalt darf nicht für kommerzielle Zwecke verwendet werden.

  Weitergabe unter gleichen Bedingungen (SA) Wenn Sie diesen Inhalt bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für einen anderen Inhalt verwenden, dann dürfen Sie den neu entstandenen Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergeben.

  Keine Bearbeitung (ND) Der Inhalt darf nicht bearbeitet oder in anderer Weise verändert werden.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieser Inhalt fällt, mitteilen.
- Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Rechtsinhabers aufgehoben werden.
- Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.
- Nachfolgend die CC-Lizenzen der Version 2.5:

  Attribution (BY)
  http://creativecommons.org/licenses/by/2.5/

  Attribution-NoDerivs (BY-ND)
  http://creativecommons.org/licenses/by-nd/2.5/

  Attribution-NonCommercial-NoDerivs (BY-NC-ND)
  http://creativecommons.org/licenses/by-nc-nd/2.5/

  Attribution-NonCommercial (BY-NC)
  http://creativecommons.org/licenses/by-nc/2.5/

  Attribution-NonCommercial-ShareAlike (BY-NC-SA)
  http://creativecommons.org/licenses/by-nc-sa/2.5/

  Attribution-ShareAlike (BY-SA)
  http://creativecommons.org/licenses/by-sa/2.5/

  Eine Übersicht befindet sich auch unter http://creativecommons.org/licenses/.

Relevanz kennt kein Verfallsdatum

Das Phänomen „Open Source" fasziniert heute mehr denn je. Dabei haben die meisten Artikel der Jahrbuch-Reihe ihre Gültigkeit bis heute behalten. Seitdem 2004 erstmals das umfassende Kompendium mit grundlegenden Erkenntnissen rund um das Prinzip der offenen Quelle zusammengetragen wurde, erfreut sich die Mischung von Beiträgen aus Wirtschaft, Verwaltung, Wissenschaft und Praxis großer Beliebtheit:

«Es macht Spaß, dieses Buch zu lesen.»
Thomas Rietig, AP-Korrespondent, über das Open Source Jahrbuch 2004

«Neben der Bündelung des fachlichen Hintergrundwissens zeigt das Buch ausgehend von Fallbeispielen auch die Potenziale in großen Unternehmen und Behörden auf. . . . Softwareentwicklung zeigt plötzlich zahlreiche rechtliche, ökonomische und sogar gesellschaftliche Facetten.»
Maik Schmidt, c't – magazin für computertechnik, Buchkritik Open Source Jahrbuch 2005

«Der Band, an dem 39 Autoren mitgewirkt haben, erhebt zu Recht den Anspruch darauf, ein aktuelles Nachschlagewerk für Wirtschaft, Verwaltung und Wissenschaft zu sein.»
Rainer Hill, Kommune21, Buchvorstellung des Open Source Jahrbuch 2006

Open Source Jahrbuch 2004, 2005 und 2006

Die komplette Reihe ist erhältlich unter http://opensourcejahrbuch.de und via Lehmanns Fachbuchhandlung.