Open Source Jahrbuch 2005 *light*

Zwischen Softwareentwicklung und Gesellschaftsmodell

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

Sie lesen die Light-Version des Open Source Jahrbuch 2005. Es sind nicht alle Beiträge enthalten, dafür sind kommerzielle Nutzung und Weitervertrieb erlaubt. Unter http://www.opensourcejahrbuch.de finden Sie den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält.

Open Source Jahrbuch 2005

Zwischen Softwareentwicklung und Gesellschaftsmodell

Herausgegeben und bearbeitet von:

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

Clemens Brandt
Student der Informatik, TU Berlin

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

Annika Held
Studentin der Medienberatung, TU Berlin

Katja Luther
Studentin der Informatik, TU Berlin

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

Wolfram Riedel
Student der Kommunikationswissenschaft
und der Informatik, TU Berlin

Patrick Stewin
Student der Informatik, TU Berlin

Sebastian Ziebell
Student der Informatik, TU Berlin

Bastian Zimmermann
Student der Informatik, TU Berlin

2005

Lehmanns Media LOB.de

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

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

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

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

Printed in the European Union.

Inhalt

Kapitel 3 – Ökonomie

Kapitel 4 – Recht und Politik

Kapitel 5 – Gesellschaft

Kapitel 6 – Open Content

Kapitel 7 – Open Innovations

Vorwort der Herausgeber

MATTHIAS BÄRWOLFF, ROBERT A. GEHRING
UND BERND LUTTERBECK

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Vorwort des Open Source Jahrbuchs 2004

ROBERT A. GEHRING UND BERND LUTTERBECK

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Synopsis

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Kapitel 1

Fallbeispiele

Open-Source-Software in geschäftskritischen Einsatzgebieten

PATRICK STEWIN

1. Einleitung zum Kapitel „Fallbeispiele“

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

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

2. Interoperabilität als Leitmotiv

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3. Hohe Anforderungen an IT-Netze

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

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

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

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

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

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

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

3.1. OSS in komplexen IT-Infrastrukturen

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4. Hemmnisse bei Migrationen

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

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

Daraus resultierend wünschen sich Behörden

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

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

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

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

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

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

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

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

bietet, kommt es auch zu Nachteilen:

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

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

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

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

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

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

9 X.500 und LDAP sind offene Standards.

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

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

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

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

5. Fachanwendungen

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

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

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

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

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

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

5.1. Open-Source-Fachanwendungen

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

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

6. Migrations-Lösungen in der Praxis

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

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

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

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

- Emulatoren
- Terminalserver
- Neuprogrammierung

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

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

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

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

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

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

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

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

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

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

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

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

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

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

3. Platz: Mohn Media – Mohndruck GmbH:

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

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

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

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

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

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

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

8. Fazit: Fortschreitende Etablierung von OSS

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

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

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

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

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

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

Literatur

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

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

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

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

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

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

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

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

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

16 Stand: 23. Januar 2005

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Die Open-Source-Strategie der öffentlichen Verwaltung

JOACHIM STURM

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

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Migration bei der BStU auf Linux-Netware/Windows XP

CHRISTIANE KUNATH

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

1. Einleitung

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

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

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

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

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

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

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

1.2. Historie der IT-Landschaft

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

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

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

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

1 Stand: September 2004

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

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

1.3. Die Notwendigkeit der Migration bei der BStU

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

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

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

Zu beachtende Besonderheiten der Migration

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

2. Ablauf der Migration

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

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

2.1. Erstellung einer Migrationsstudie

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

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

- Ist-Aufnahme
- Infrastrukturalternativen
- Nutzwertanalyse
- Wirtschaftlichkeitsbetrachtung

Ist-Aufnahme

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

Infrastrukturalternativen

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

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

2 Stand: September 2004

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

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

Nutzwertanalyse (Qualitative Bewertung)

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

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

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

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

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

IT-Wirtschaftlichkeitsbetrachtung (IT-WiBe)

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

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

3 Visual Basic wird unter Linux nicht unterstützt.

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

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

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

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

Zusammenfassung des Ergebnisses der Migrationsstudie

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- Groupware
- Verzeichnisdienst
- Fachanwendungen

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

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

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

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

5 Dafür wurde ein externer Dienstleister beauftragt.

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

2.3. Die Migration im Jahr 2004

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

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

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

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

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

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

7 Alle Server in der DMZ sind Linux-Server.

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

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

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

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

3. Fazit und Ausblick

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

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

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

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

8 Stand: September 2004

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

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

Literatur

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

Weiterführende Informationen

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

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

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

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

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

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

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

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

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

Die Schriftstücke der KBSt sind unter den folgenden Adressen abrufbar:

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

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

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

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

Linux im Rathaus – Ein Migrationsprojekt der Stadt Schwäbisch Hall

HORST BRÄUNER

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

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Migration auf Samba/OpenLDAP bei der Norddeutschen Affinerie AG

JÖRG MEYER UND CARSTEN BRUNKE

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

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

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

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

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

2. Entscheidungsvorbereitung

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

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

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

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

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

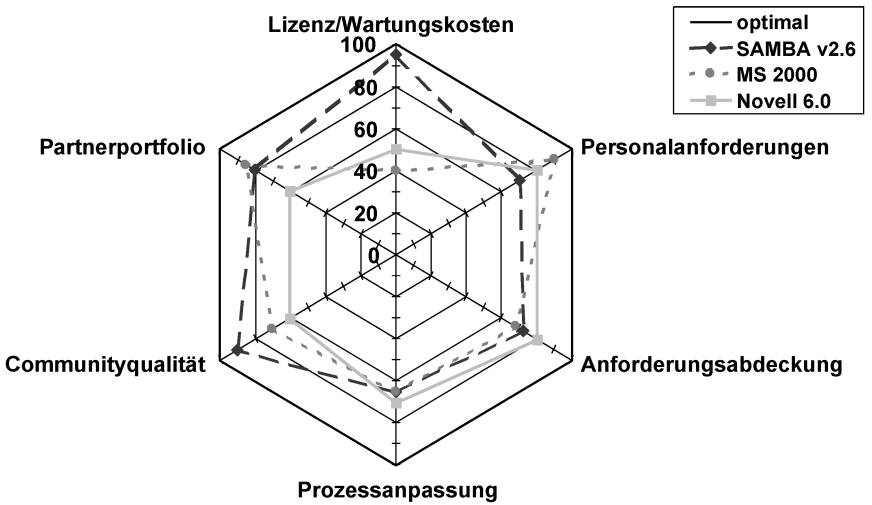

Abbildung 1: Softwareanalyse

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

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

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

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

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

3. Evaluierungsphase

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

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

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

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

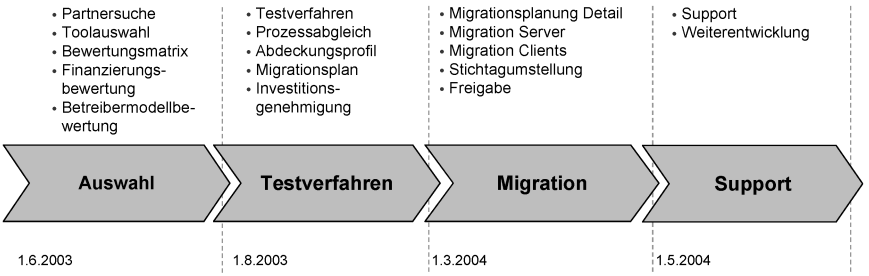

Abbildung 2: Migrationsfahrplan, Ziel: integrierte und kostengünstige File/Print-Lösung

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

4. Vorbereitung

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

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

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

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

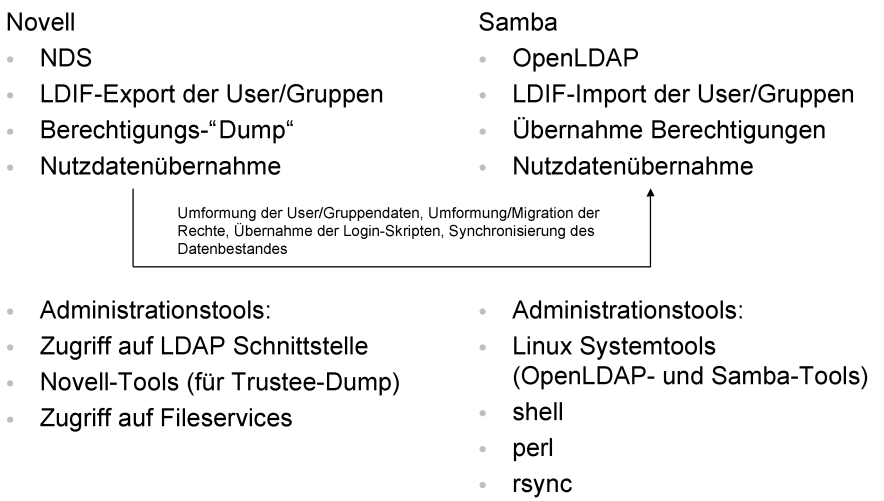

Abbildung 3: Migrationspfade

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

5. Umstellung

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

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

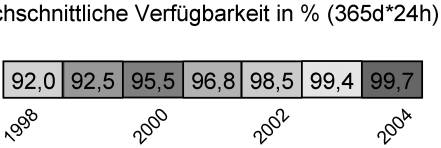

Abbildung 4: Entwicklung der NA-IT in Zahlen I

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

6. Technisches Ergebnis

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

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

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

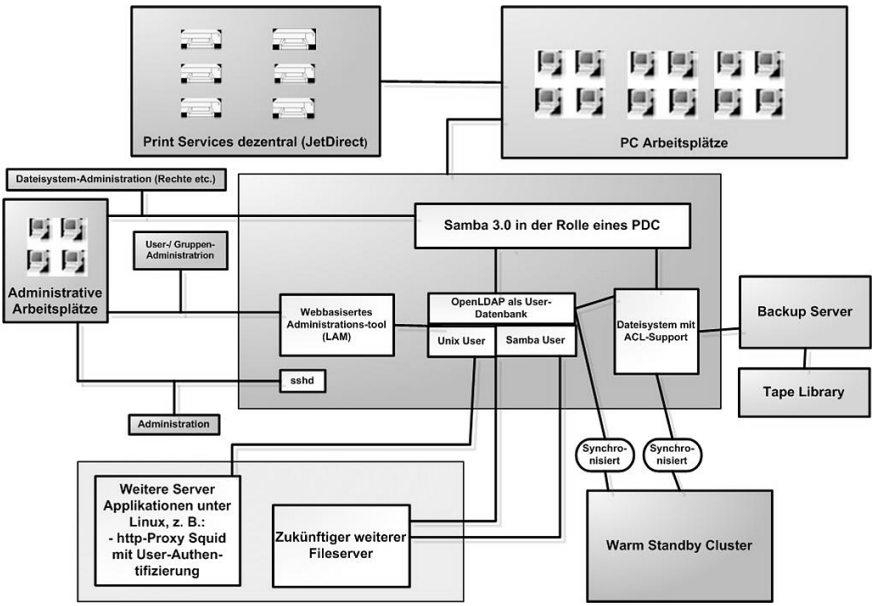

Abbildung 5: File/Print-Dienste nach der Umstellung

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

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

7. Ganzheitliches Ergebnis

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

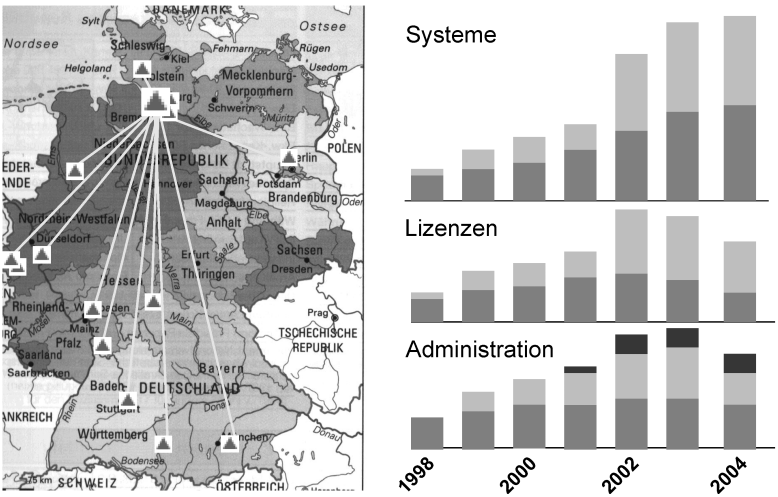

Abbildung 6: Entwicklung der NA-IT in Zahlen II

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

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

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

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

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

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

8. Ausblick und Ableitungen für eine OSS-Gesamtstrategie

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

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

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

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

GENOMatch – Datenschutz für die pharmakogenetische Forschung

Broder Schümann und Denis Petrov

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

1. Einleitung – Das GENOMatch-Projekt

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

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

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

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

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

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

2. Überblick – Pharmakogenetische Forschung

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

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

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

3. Datenschutz – Erfordernisse und Datenschutzkonzept

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

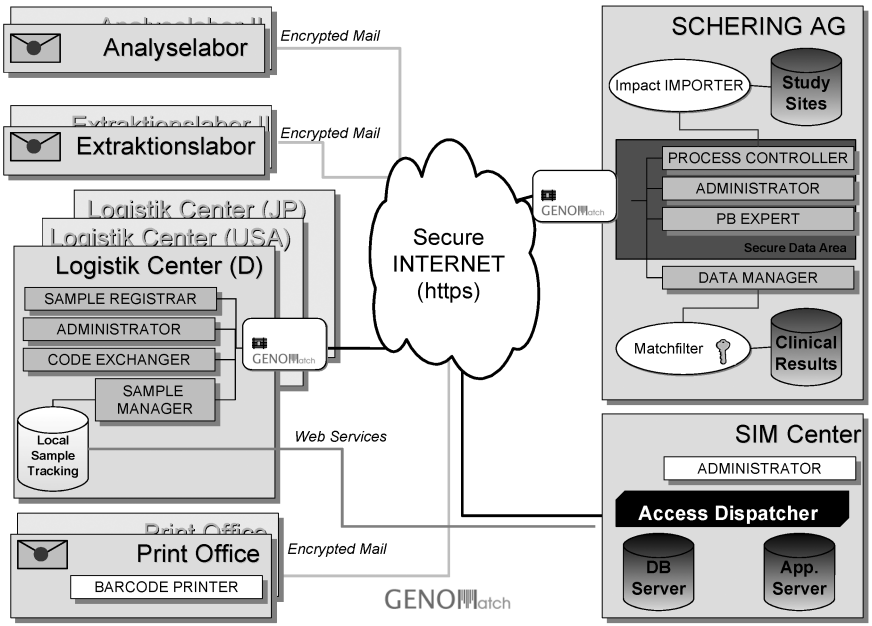

Abbildung 1: GENOMatch-B2B-Interaktionen

Council of Europe (1997) dazu:

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

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

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

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

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

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

4. GENOMatch-*Workflow* – Doppelte Pseudonymisierung

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

An dem GENOMatch-Prozess nehmen folgende Institutionen teil:

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

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

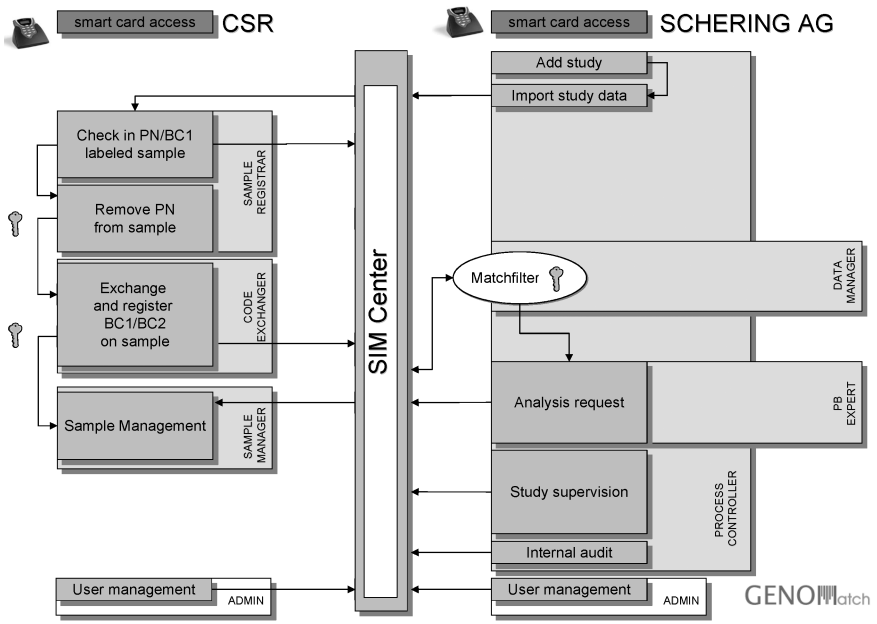

Abbildung 2: GENOMatch-Abläufe

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

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

5. IT-Unterstützung – Technischer Datenschutz

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

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

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

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

6. Open Source – Entscheidung und Erfahrungen

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

6.1. Architektur

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

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

Tabelle 1: Betrachtete Standardumgebungen

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

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

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

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

6.2. Server-Seite – SIM-Center

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

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

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

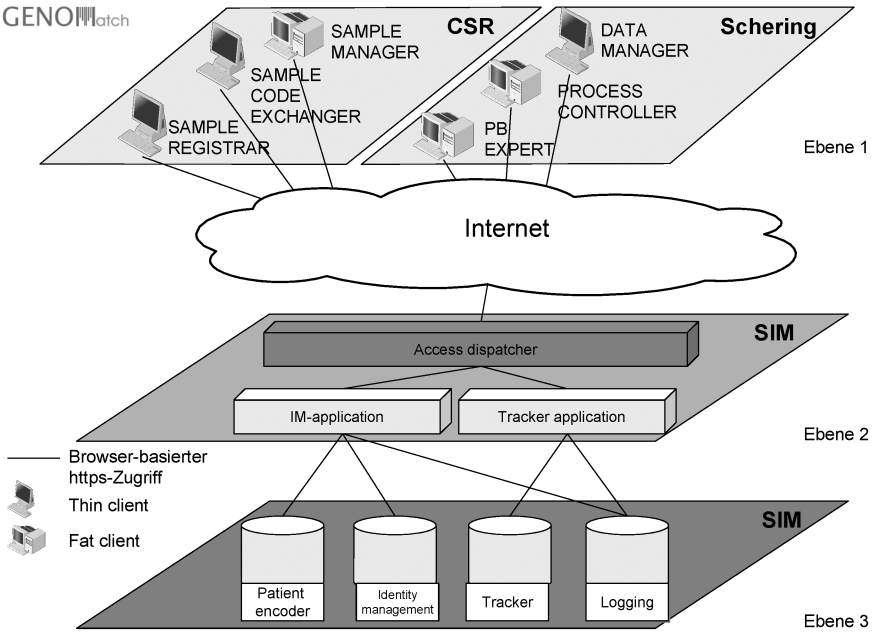

Abbildung 3: GENOMatch-Architektur

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

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

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

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

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

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

6.3. Thin Clients

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

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

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

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

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

6.4. Smartcards

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

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

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

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

Dadurch wird die Sicherheit in mehreren Bereichen verbessert:

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

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

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

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

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

7. Fazit

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

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

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

Literatur

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

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

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

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

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

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

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

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

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

Weiterführende Informationen

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

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

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

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

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

Kapitel 2

Technik

Open Code Worlds

WOLFRAM RIEDEL

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Open Source und Usability

JAN MÜHLIG

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

1. Einleitung

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

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

1 SuSE-Distribution mit KDE-Arbeitsumgebung

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

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

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

2. Entwickeln für die Community

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

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

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

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

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

3. Usability ist einfach?

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

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

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

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

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

4. OSS-Usability = Bazaar?

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

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

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

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

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

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

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

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

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

5. Fehlende Ressourcen

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

6. Geschwindigkeit als zentraler Vorteil

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

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

7. OpenUsability

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

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

8. Neue Ressourcen

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

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

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

9. Die Zukunft

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

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

Literatur

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

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

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

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

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

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

Der Beitrag freier Software zur Software-Evolution

ANDREAS BAUER UND MARKUS PIZKA

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

1. Einführung

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

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

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

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

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

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

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

2. Open Source und Freie Software

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

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

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

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

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

2.1. Mythen und Klischees

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

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

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

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

2.2. Wirklichkeit

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

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

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

2.3. Verwandte Arbeiten zur Software-Evolution

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

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

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

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

3. Evolution des Entwicklungsprozesses

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

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

3.1. Unausweichliche technische Veränderungen

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

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

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

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

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

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

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

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

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

3.2. Änderung der Organisation

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

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

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

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

Tabelle 2: Führungsgremien in freien Softwareprojekten

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

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

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

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

3.3. Konfigurations- und Änderungsmanagement

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

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

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

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

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

4. Evolution der Architektur

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

4.1. Modulare Programme und Schichten

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

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

4.2. Verstrickung von Prozess und Architektur

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

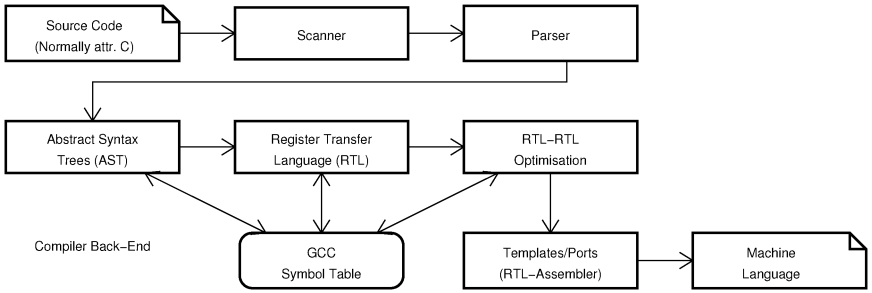

Abbildung 1: Logische und physische Module des GCC Kerns

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

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

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

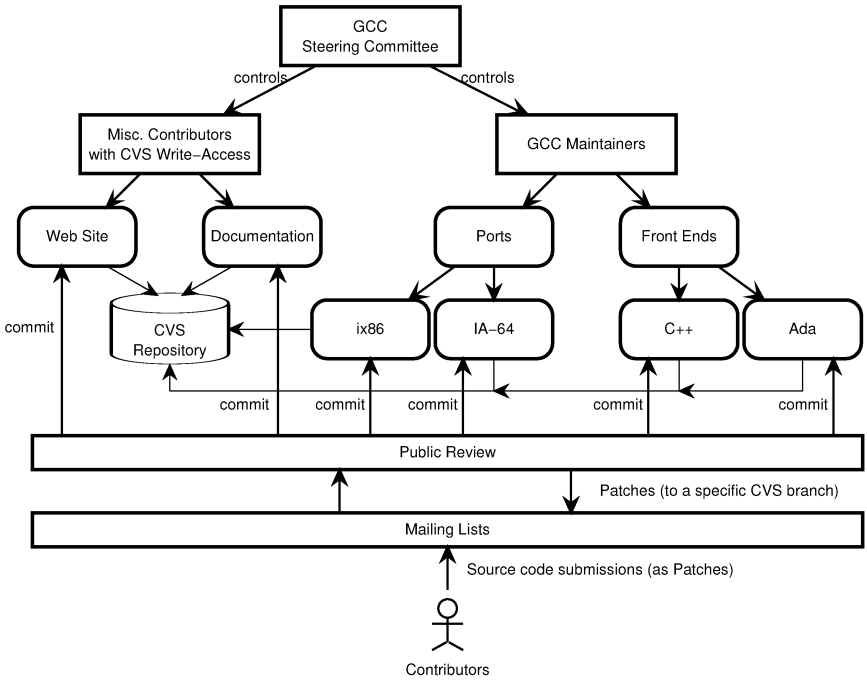

Abbildung 2: Die wesentlichen Elemente der GCC Projektstruktur und Organisation

4.3. Projekt- und Interprojekt-Abhängigkeiten

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

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

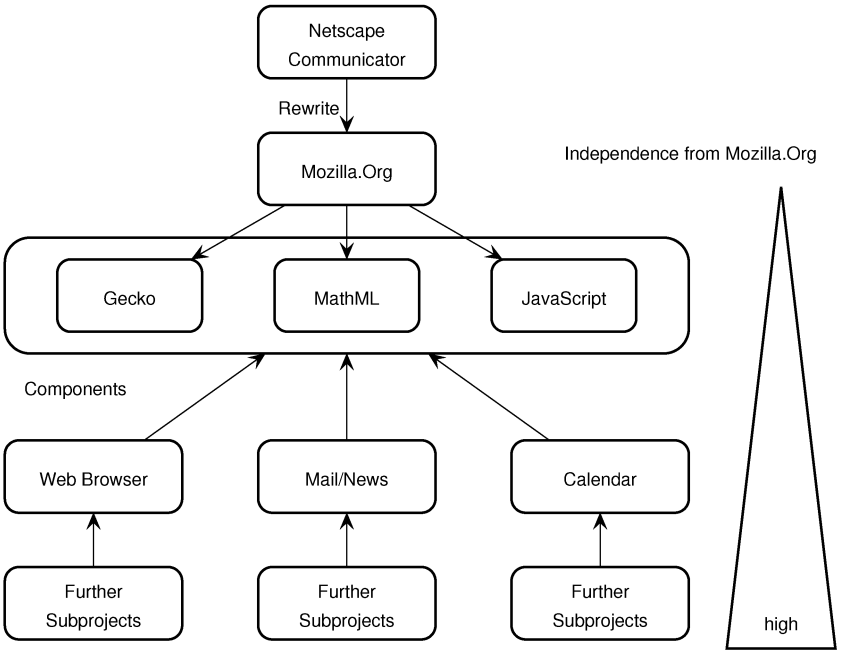

Abbildung 3: Interprojekt-Abhängigkeiten im Mozilla-Projekt

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

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

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

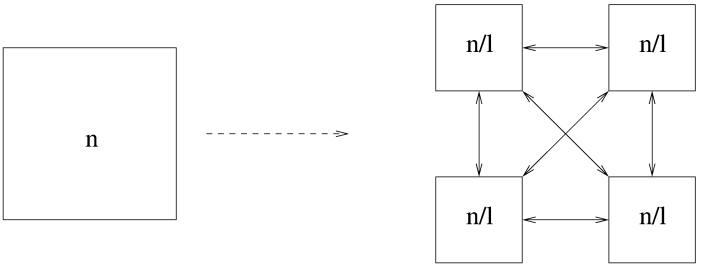

Abbildung 4: Code-Splitting

4.4. Komplexitätsreduktion durch Code-Aufteilung

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

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

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

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

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

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

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

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

4.5. Inkrementelle Überarbeitungen

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

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

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

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

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

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

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

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

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

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

5. Fazit

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

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

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

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

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

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

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

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

Bekanntmachung

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

Literatur

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Snowbox

OLIVER FEILER

(GPL 2.0 siehe Seite 464)


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

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

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

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

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

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

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

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

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

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


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

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

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


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

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


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

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

```
sub pop_cmd_list {
        my($message) = shift;
        if (!defined($message)) {
                print "+OK $num_messages messages
                  ($maildrop_size octets)\r\n";
                foreach (@maildrop) {
                        # Do not include deleted messages in listing
                        if (%$_->{"deleted"} == 0) {
                                print %$_->{"num"}." ".%$_->{"len"}."\r\n";
                        }
                }
                print ".\r\n";
        } else {
                $message =~ tr/0-9//cd;          # Must only contain 0-9
                if (check_valid_msg_num($message) == -1) {
                        return;
                }
                if (($message <= scalar(@maildrop)) &&
                    ($maildrop[$message-1]->{"deleted"} == 0)) {
                        print "+OK $message "
                          .$maildrop[$message-1]->{"len"}."\r\n";
                } else {
                        print_err_no_such_msg();
                }
        }
}

sub pop_cmd_noop {
        print "+OK *sighs idly*\r\n";
}

sub pop_cmd_quit {
        my($logged_in) = shift;
        if ($logged_in) {
                # Write maildrop
                seek (MAILDROP, 0, 0);
                truncate (MAILDROP, 0);
                foreach (@maildrop) {
                        if (%$_->{"deleted"} == 0) {
                                print MAILDROP %$_->{"body"};
                        }
                }
                nuke_locks();
        }
        print "+OK POP3 server signing off. Have a nice day.\r\n";
        closelog();                              # Close the syslog
        exit(0);
}
```

```
sub pop_cmd_retr {
        my($message) = shift;
        my($msg);
        if (!defined($message)) {
                print_err_no_such_msg();
                return;
        }
        $message =~ tr/0-9//cd;                 # Must only contain 0-9
        if (check_valid_msg_num($message) == -1) {
                return;
        }
        if ($message <= scalar(@maildrop)) {
                if ($maildrop[$message-1]->{"deleted"} == 1) {
                        print "-ERR message was deleted.\r\n";
                        return;
                }
        }
        if ($message <= scalar(@maildrop)) {
                print "+OK ".$maildrop[$message-1]->{"len"}." octets\r\n";
                $msg = $maildrop[$message-1]->{"body"};
                $msg =~ s/\n/\r\n/g;      # Replace all line endings with CRLF
                $msg =~ s/\r\n\.(?=\r\n)/\r\n\.\./g; # Prepad .CRLF -> ..CRLF
                print $msg;
                print ".\r\n";
        } else {
                print_err_no_such_msg();
        }
}

sub pop_cmd_stat {
        print "+OK $num_messages $maildrop_size\r\n";
}


sub pop_cmd_uidl {
        my($message) = shift;
        if (!defined($message)) {
                print "+OK Just a second, UIDL listing follows.\r\n";
                foreach (@maildrop) {
                        if (%$_->{"deleted"} == 0) {
                                print %$_->{"num"}." "
                                  .md5_hex(%$_->{"body"})."\r\n";
                        }
                }
                print ".\r\n";
        } else {
                $message =~ tr/0-9//cd;         # Must only contain 0-9
                if (check_valid_msg_num($message) == -1) {
                        return;
                }
                if (($message <= scalar(@maildrop)) &&
                    ($maildrop[$message-1]->{"deleted"} == 0)) {
                        print "+OK ".$maildrop[$message-1]->{"num"}." "
                          .md5_hex($maildrop[$message-1]->{"body"})."\r\n";
                } else {
                        print_err_no_such_msg();
                }
        }
}
```

```
sub pop_cmd_unknown {
        print "-ERR I have no idea what you want from me.\r\n";
        if ($loglevel >= 2) {
                syslog('info', "connection from $remote:
                  client sent unknown command.");
        }
}


sub check_valid_msg_num {
        my($message) = shift;
        if (($message eq "") || ($message < 1)) {
                print_err_no_such_msg();
                return -1;
        }
        return 0;
}


sub pop_cmd_not_logged_in {
        print "-ERR You are not logged in.\r\n";
        if ($loglevel >= 2) {
                syslog('info', "connection from $remote:
                  client sent transaction command while not logged in.");
        }
}

sub pop_cmd_too_long {
        print "-ERR Command too long.\r\n";
        if ($loglevel >= 1) {
                syslog('warning', "connection from $remote:
                  command sent was too long.");
        }
}


sub print_err_no_such_msg {
        print "-ERR No such message.\r\n";
}


sub acquire_locks {
        my($mtime);
        if (!flock (MAILDROP, LOCK_EX | LOCK_NB)) {
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          could not lock mailbox for \'$user\' (flock).");
                }
                die "-ERR Mailbox is locked.
                  Is another POP3 session active?\r\n";
        }
        # see if there is a dotlock. try to delete it if older than 15 mins
        if (-e"$maildrops/$user.lock") {
                $mtime = (stat("$maildrops/$user.lock"))[9];
                if (time()-$mtime > 900) {
                        if ($loglevel >= 1) {
                                syslog('warning', "connection from $remote:
                                  dotlock for \'$user\' is older than 15 min,
                                  ignoring.");
                        }
                        unlink ("$maildrops/$user.lock");
                }
        }
```

```
        # also set a dotlock here in case other programs ignore flock.
        if (!sysopen (DOTLOCK,
              "$maildrops/$user.lock", O_CREAT | O_RDWR | O_EXCL)) {
                if ($loglevel >= 1) {
                        syslog('warning', "connection from $remote:
                          could not lock mailbox for \'$user\' (dotlock).");
                }
                die "-ERR Mailbox is locked.
                  Is another POP3 session active?\r\n";
        }
        print DOTLOCK $PID;
        close (DOTLOCK);
}


sub nuke_locks {
        flock (MAILDROP, LOCK_UN);
        unlink ("$maildrops/$user.lock");
        close (MAILDROP);
}

# Remove the locks if we get killed from inetd (SIGPIPE?)
sub signal_handler {
        if ($logged_in) {
                nuke_locks();
                if ($loglevel >= 2) {
                        syslog('info', "connection from $remote:
                          connection aborted, removing locks.");
                }
        } else {
                if ($loglevel >= 2) {
                        syslog('info', "connection from $remote:
                          connection aborted.");
                }
        }
        exit(2);
}


sub config {
        open (CONFIG, "$snowbox_config") || return; # and use default config
        while (<CONFIG>) {
                chomp;
                if (/^#/) { next; }
                /^(.*?)\s+(.*)$/;
                if ($1 eq "authfile") {
                        $passwordfile = $2;
                } elsif ($1 eq "maildir") {
                        $maildrops = $2;
                } elsif ($1 eq "maildir_gid") {
                        $maildrop_gid = $2;
                } elsif ($1 eq "loglevel") {
                        $loglevel = $2;
                }
        }
        close (CONFIG);
}
```

Die industrielle Nutzung und Entwicklung von Open-Source-Software: Embedded Linux

JOACHIM HENKEL UND MARK TINS

(CC-Lizenz siehe Seite 463)

Unternehmen tragen in stark steigendem Maße zur Entwicklung von Open–Source-Software (OSS) bei. Einen besonderen Fall von kommerzieller OSS–Entwicklung stellt „Embedded Linux" dar, also Varianten von Linux, die an den Gebrauch in „eingebetteten Systemen", wie z. B. Maschinensteuerungen und Handys, angepasst sind. Während Unternehmen wie IBM und Sun OSS aus strategischen Gründen und als Komplement zu ihren Produkten unterstützen, stellt *Embedded Linux* für Gerätehersteller einen festen Bestandteil ihres Produktes dar, für spezialisierte Softwareunternehmen sogar ihr Kerngeschäft. Daher wird die Entscheidung, eigene Entwicklungen zu *Embedded Linux* öffentlich zu machen, wesentlich vorsichtiger und selektiv getroffen. Es wird beschrieben, welche Schutzmöglichkeiten Unternehmen trotz der General Public License zur Verfügung stehen und in welchem Ausmaß Code trotz dieser Schutzmöglichkeiten freigegeben wird. Eine Analyse der Motive für eine Freigabe zeigt, dass externe Entwicklungsunterstützung tatsächlich als wichtiger Vorteil wahrgenommen wird. Für Softwareunternehmen, vor allem kleinere, spielt zudem der Marketingaspekt eine Rolle. Als begünstigend für OSS-Prozesse erweist sich die Tatsache, dass der Technologiebedarf im Bereich *Embedded Linux* extrem heterogen ist. Reines Trittbrettfahren ist daher kaum möglich; vielmehr müssen Unternehmen fast immer gewisse Anpassungen und Weiterentwicklungen bestehenden Codes vornehmen. Die rapide Verbreitung von *Embedded Linux* hat Konsequenzen für die gesamte Branchenstruktur der eingebetteten Betriebssysteme. Zum einen wird auch auf Anbieter proprietärer Betriebssysteme Druck ausgeübt, ihren Kunden den Quellcode verfügbar zu machen. Zum anderen reduziert *Embedded Linux* die Markteintrittsbarrieren, so dass eine zunehmende Fragmentierung des Marktes zu erwarten ist. Zusammenfassend lässt sich feststellen, dass trotz des weitgehend kommerziellen Umfeldes – Hobby-Entwickler spielen nur eine sehr untergeordnete Rolle – OSS-Entwicklungsprozesse im Bereich *Embedded Linux* sehr gut funktionieren.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Kapitel 3

Ökonomie

Die ökonomischen Dimensionen von Open Source

MATTHIAS BÄRWOLFF

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

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

JENS MUNDHENKE

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

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Open-Source-Software und Standardisierung

CHRISTIAN MAASS UND EWALD SCHERM

Open-Source-Software steht seit geraumer Zeit im öffentlichen Interesse. Die Auseinandersetzung mit Fragen der Standardisierung erfolgt allerdings nur am Rande. Vielmehr werden damit zusammenhängende Probleme nur oberflächlich dargestellt, oder es kommt zu Fehleinschätzungen. So erachtet man OSS und offene Standards mitunter als Synonyme, obwohl es sich dabei um zwei grundlegend verschiedene Sachverhalte handelt. Vor diesem Hintergrund erfolgt in diesem Beitrag eine Auseinandersetzung mit drei Aspekten, die in Zusammenhang mit Fragen der Standardisierung zwar immer wieder angesprochen, aber nur oberflächlich dargestellt werden: (1) Kompatibilitätsprobleme in Folge des *Forking*, (2) Hersteller(un)abhängigkeit und (3) der Einfluss von Softwarepatenten. Die Auseinandersetzung mit diesen Aspekten soll dazu beitragen, die Diskussion um Open-Source-Software und damit einhergehende Fragen der Standardisierung, differenzierter führen zu können.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Open-Source-Software als Signal

MAIK HETMANK

Dieser Beitrag befasst sich mit dem Aspekt des Signalerwerbs als Motivation zur Entwicklung von Open-Source-Software. Da Programmierkenntnisse zumeist sehr spezifisch und nicht direkt beobachtbar sind, kann es sinnvoll sein, diese Information mittelbar durch die Mitarbeit an Open-Source-Software-Projekten zu kommunizieren. Der Open-Source-Entwicklungsprozess gewährleistet die dauerhafte Sichtbarkeit sowie die Glaubwürdigkeit von solchen Signalen. Jüngere Untersuchungen zeigen empirisch, dass Signalerwerb ein wichtiges Motiv zur Mitarbeit bei Open-Source-Software darstellt. Diese Studien belegen auch, dass die Mitwirkung an Open-Source-Projekten, speziell bei sehr guten Programmierern, konkrete Auswirkungen auf das erzielte Lohnniveau haben kann.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Das Microsoft-Shared-Source-Programm aus der Business-Perspektive

WALTER SEEMAYER UND JASON MATUSOW

Dieser Artikel beschreibt und begründet die strategischen Erwägungen hinter der Shared-Source-Initiative von Microsoft. Hierfür wird der Begriff des Software-Ökosystems eingeführt, der die wechselseitigen Abhängigkeiten von kommerziellen Firmen und nicht-kommerziellen Institutionen wie Universitäten beschreibt. Die Shared-Source-Initiative versucht, einige der unbestrittenen Vorteile von Open-Source-Entwicklung zu übernehmen und mit dem Geschäftsmodell von Microsoft zu vereinbaren, dass traditionell auf der Wertschöpfung durch Lizenzierung von proprietären Softwareprodukten basiert.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Coases Pinguin beginnt zu fliegen – Der institutionelle Wandel in der Softwareindustrie

MATTHIAS BÄRWOLFF

Dieser Artikel behandelt den Wandel der Prozesse und Strukturen in der kommerziellen Softwareindustrie bedingt durch den wachsenden Einfluss von Open Source als „Produktionsparadigma“. Der zu beobachtende institutionelle Wandel lässt sich als effizienzsteigernde Anpassung der Produktions- und Nutzungsstrukturen im Bereich Software begreifen.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Kapitel 4

Recht und Politik

Von Lizenzen und Patenten

KATJA LUTHER

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Quelloffene Software auf Ebene der Europäischen Gemeinschaft

ANDREAS NEUMANN[1]

Software ist seit jeher ein kaum an nationale Grenzen gebundenes Gut, die nationalstaatliche Steuerungsfähigkeit auf diesem Gebiet entsprechend eingeschränkt. Umso bedeutsamer sind für die betreffenden Märkte und Akteure die Ziele und Initiativen inter- und supranationaler Organisationen wie insbesondere auch der Europäischen Gemeinschaft (EG). Der Beitrag schildert die – bislang noch im Wesentlichen fragmentarischen – Ansätze der EG mit Bezug auf quelloffene Software (Open-Source-Software) und die gegenwärtige Tätigkeit der EG auf diesem Gebiet. Dabei dienen die Maßahmen zur Interoperabilitätsfänderung und die Richtlinie zur Patentierbarkeit computerimplementierter Erfindungen (Softwarepatente) als Referenzpunkte auf Mikroebene vor dem Hintergrund der Makroebene des Primärrechts.

1. Einleitung

Die fortschreitende europäische Integration hat eine Verlagerung politischer Entscheidungsprozesse von der nationalen auf die Gemeinschaftsebene bewirkt. Dieser Befund ist zwar alles andere als neu, aber von zentraler Bedeutung für das Verständnis realpolitischer Entscheidungsprozesse.

Auf der Ebene der Mitgliedstaaten wird zunehmend nur noch umgesetzt, was zuvor auf der Ebene der Europäischen Gemeinschaft beschlossen wurde. Damit ist nicht unbedingt ein Bedeutungsverlust nationaler Politiksetzungsfähigkeit, aber jedenfalls eine Verschiebung der Einflussnahmemöglichkeiten zwischen den einzelnen mitgliedstaatlichen Staatsgewalten verbunden. Das zentrale Gesetzgebungsorgan der Gemeinschaft, dessen Zusammensetzung unmittelbar die politischen Machtverhältnisse auf mitgliedstaatlicher Ebene widerspiegelt, ist natürlich der Rat – und dieser setzt sich aus Vertretern der Mitgliedstaaten auf Ministerebene zusammen.[2] Der mitgliedstaatliche Gesetzgeber wird also nicht mehr nur rein faktisch (aufgrund der überlegenen Ressourcen der Ministerialbürokratie), sondern – über den Umweg über Europa – auch rechtlich im Wesentlichen durch die mitgliedstaatliche Exekutive vorgesteuert.

1 Der Verfasser dankt Frau *Julia Wetzel* für wertvolle Vorarbeiten zu diesem Beitrag.

2 Art. 203 Abs. 1 EG-Vertrag.

Diese Tendenzen müssen zwar (zu Recht) demokratietheoretische Bedenken wecken. Sie scheinen aber hierzulande dem insoweit der Verantwortung zu eigenständigen Entscheidungen enthobenen mitgliedstaatlichen Gesetzgeber sogar durchaus gelegen zu sein. Dies zeigte jüngst beispielsweise das Gesetzgebungsverfahren zum neuen Telekommunikationsgesetz deutlich, in dem der Deutsche Bundestag auch auf Druck der Kommission der Europäischen Gemeinschaften ohne (gemeinschafts-) rechtliche Not die eigene Steuerungsfähigkeit beschränkte und sich statt der noch seitens der Bundesregierung vorgesehenen sinnvollen Formulierung eigener regulierungspolitischer Zielsetzungen mit der bloßen Übernahme untergesetzlicher Empfehlungskriterien der Kommission begnügte.[3] In jedem Fall verdeutlichen die geschilderten Entwicklungen, dass die nationale Steuerungsfähigkeit auf der Gemeinschaftsebene rechtlich wie politisch überformt ist. Für praktisch alle Politikbereiche jenseits des aus staatstheoretischen Gründen für die Mitgliedstaaten reservierten Bereiches (*domaine réservé*) ist der „Blick nach Brüssel" bzw. in die anderen Kapitalen der Europäischen Gemeinschaft daher nicht nur lohnend, sondern unverzichtbar. Dies gilt damit auch für den Gegenstand des vorliegenden Sammelwerkes, also für die Entwicklung im Bereich der quelloffenen Software. Ausgangspunkt soll dabei auf der Makroebene eine Analyse der Kompatibilität des Konzepts quelloffener Software mit den grundlegenden Zielen und Aufgaben der Europäischen Gemeinschaft sein. Danach werden dann auf der Mikroebene die beiden Politikbereiche dargestellt, in denen quelloffener Software die bislang bedeutendste Rolle auf Gemeinschaftsebene zukommt.

2. Die Makroebene: quelloffene Software als Herausforderung für das EG-Primärrecht

Jenseits des aktuellen Verfassungsgebungsprozesses verfügt die Europäische Gemeinschaft schon seit jeher über eine „Verfassung" im rechtstechnischen Sinne – den (seitdem mehrfach geänderten) Gründungsvertrag aus dem Jahre 1957. Er „verfasst" die Gemeinschaft. Jedes Handeln ihrer Organe muss letzten Endes in ihm seine Legitimationsgrundlage finden, Maßnahmen der Gemeinschaft, die gegen den Vertrag verstoßen, sind rechtswidrig. Dieser Vorrang des EG-Vertrages auch vor anderen Rechtsakten auf Gemeinschaftsebene findet seinen Ausdruck in der Bezeichnung „Primärrecht". Darunter versteht man generell die Gründungsverträge der Europäischen Gemeinschaft (EG), der (praktisch nur noch wenig bedeutsamen) Europäischen Atomgemeinschaft (EAG) und der mittlerweile aufgelösten Europäischen Gemeinschaft für Kohle und Stahl (EGKS), jeweils einschließlich etwaiger Anlagen, Anhänge und Protokolle sowie späterer Ergänzungen und Änderungen. (Koenig und Haratsch 2003, S. 3) Die von Organen dieser Gemeinschaften erlassenen Rechtsakte bezeichnet man demgegenüber als „Sekundärrecht" (Koenig und Haratsch 2003, S. 3) bzw. entsprechend dem oben dargestellten Legitimationszusammenhang als „abgeleitetes

3 Zur Diskussion um das damit in Bezug genommene Regulierungsziel des „funktionsfähigen Wettbewerbs" und der Zentralnorm des § 10 Abs. 2 des neuen Telekommunikationsgesetzes ausführlich Loetz und Neumann (2003).

Recht". In welchen Bereichen und mit welcher Zielsetzung die Gemeinschaft, womit hier und im Folgenden nur noch die EG gemeint sein soll, tätig wird bzw. überhaupt nur tätig werden darf, richtet sich daher letzten Endes nach der Reichweite des EG-Primärrechts.

Die damit auch für nachrangige Maßnahmen maßgebliche Aufgabe der EG wird in Art. 2 des EG-Vertrags definiert, wobei zugleich die Mittel zur Erfüllung dieser Aufgabe festgelegt werden. Die Aufgabe ist auf Förderung gerichtet – dies betrifft zum einen allgemeine soziale Ziele, wie z. B. das der „Gleichstellung von Männern und Frauen", „ein hohes Maß an Umweltschutz und Verbesserung der Umweltqualität" sowie zum anderen allgemein „die Hebung der Lebenshaltung und der Lebensqualität". Bereits der Aufgabenkatalog des Art. 2 macht aber deutlich, dass die Europäische Gemeinschaft (ursprünglich, aber eben auch noch nach der Neubezeichnung der „Europäischen Wirtschaftsgemeinschaft" [EWG] als „Europäische Gemeinschaft" im Jahre 1993) primär eine Wirtschaftsgemeinschaft ist. So gehört zu den in Art. 2 ausdrücklich genannten Aufgaben der Gemeinschaft nämlich gerade auch, die „Entwicklung des Wirtschaftslebens", „ein hohes Beschäftigungsniveau", „ein beständiges, nichtinflationäres Wachstum" sowie „einen hohen Grad von Wettbewerbsfähigkeit und Konvergenz der Wirtschaftsleistungen" zu fördern.

Diese Priorität des Wirtschaftlichen wird auch in der ebenfalls in Art. 2 enthaltenen Festschreibung der Mittel deutlich, die der Gemeinschaft zur Erreichung ihrer Ziele zur Verfügung stehen: Neben den in Bezug genommenen gemeinsamen Politiken und Maßnahmen nach Art. 3 und 4 des EG-Vertrags wird die Gemeinschaft nämlich in erster Linie „durch die Errichtung eines Gemeinsamen Marktes und einer Wirtschafts- und Währungsunion" tätig. Damit wird auch auf instrumentaler Ebene der Gedanke einer Wirtschaftsgemeinschaft betont. Dabei nimmt der EG-Vertrag nicht Bezug auf eine beliebige Wirtschaftsform, sondern bekennt sich – mit Ausnahmen für bestimmte Bereiche – ausdrücklich zur Marktwirtschaft (Vgl. Koenig und Haratsch 2003, S. 3, Rn. 8). Hinzu kommt die Nennung des Gemeinsamen Marktes als besonders hervorgehobenes Mittel zur Erreichung der Ziele der Gemeinschaft.[4] Der EG-Vertrag nimmt also den Markt als ein fundamentales Konzept der ökonomischen Theorie in Bezug. Unter einem Markt versteht man denjenigen Ort, an dem Angebot und Nachfrage nach einem bestimmten Gut, also nach einer Ware oder Dienstleistung, aufeinander treffen (Koenig et al. 2004, S. 37). Zentraler Koordinationsparameter der Güterallokation ist dabei der Preis, so dass das Konzept des Marktes letztlich auf der Idee des entgeltlichen Güteraustausches beruht.[5]

Daraus wird unmittelbar ersichtlich, dass quelloffene Software nur schwer in den konzeptionellen Ansatz einzupassen ist, auf dem die wirtschaftspolitische Fundierung der Gemeinschaft aufbaut. Denn quelloffene Software wird von ihren Produzenten –

4 Siehe dazu EuGH, Slg. 1982, 1409, 1431 f., Tz. 33; Ukrow in: Calliess und Ruffert (2002), Art. 2 EG-Vertrag Rn. 26 und 30.

5 Vgl. auch den expliziten Hinweis auf die Entgeltlichkeit der in Rede stehenden Betätigung als Wesensmerkmal für die Zurechnung zum „Wirtschaftsleben" bei EuGH, Slg. 1974, 1405, 1418, Tz. 4/10; Slg. 1976, 1333, 1340, Tz. 12/13; Slg. 1988, 6159, 6173, Tz. 12; Ukrow in: Calliess und Ruffert (2002), Art. 2 EG-Vertrag Rn. 14.

zwar nicht notwendigerweise, aber doch in der ganz überwiegenden Zahl der Fälle – unentgeltlich (frei) zur Verfügung gestellt.[6] Der (in Geld messbare[7]) Preis spielt als Koordinationsparameter keine Rolle, der Mechanismus des Marktes im herkömmlichen Sinne ist außer Kraft gesetzt. Auf derartige, im unmittelbaren Bezug kostenlose Güter ist der Marktansatz des Primärrechts nicht zugeschnitten. Wo es schon keinen Markt (im auf Austausch gerichteten Sinne des EG-Primärrechts) geben kann, kann es erst recht keinen Gemeinsamen Markt geben. Freie quelloffene Software passt nicht in das Konzept einer (Markt-) Wirtschafts- und Währungsunion.[8]

Allerdings hat natürlich auch quelloffene Software (markt-) wirtschaftliche Auswirkungen. So ist sie einerseits in der Lage, eine ansonsten durch entgeltliche Güter gedeckte Nachfrage zu befriedigen: Erfolgreiche Open-Source-Projekte, wie das Betriebssystem Linux, der Web-Browser Firefox oder die Bürobedarfslösung OpenOffice haben bewiesen, dass sie in der Lage sind, kommerzielle Software mit (ungefähr) entsprechender Funktionalität zu ersetzen – damit verringert quelloffene Software das Volumen der (auf den Austausch von Gütern und Geld gerichteten) jeweiligen Märkte. Auf diese Weise dient sie nicht dem Auf-, sondern gewissermaßen dem Abbau des (gemeinsamen) Marktes und läuft insoweit dem ökonomischen Ansatz des EG-Primärrechts diametral zuwider.

Andererseits ist freie quelloffene Software aber auch Gegenstand von Austauschverhältnissen und somit in marktwirtschaftliche Zusammenhänge im Sinne des EG–Primärrechts eingebunden.[9] Freilich sind diese Zusammenhänge bei quelloffener Software erheblich subtiler als bei herkömmlichen Gütern, da sie zwischen ihrem Anbieter und dem Nachfrager gerade nicht entgeltlich gehandelt wird. Stattdessen erfolgen (markt-) wirtschaftliche Austauschverhältnisse, die durch quelloffene Software gefördert werden, auf Ebenen, die der Stufe des Angebots vor- und nachgelagert sind. So gehen zunehmend Unternehmen der Software- und Hardwareindustrie dazu über, bestimmte Programme – zum Beispiel Treibersoftware – der Allgemeinheit (und insbesondere den Kunden ihrer sonstigen Produkte) als freie quelloffene Software anzubieten.[10] Bereits im Jahr 2001 gaben ca. 13 % aller (befragten) Unternehmen, deren Hauptzweck die Entwicklung von Software ist, an, zumindest gelegentlich quelloffene Software anzubieten (vgl. Blind et al. 2001, S. III). Diese quelloffene Software wird also von bezahlten Programmierern entwickelt (Koenig und Neumann (2004*b*, S. 130),

6 Etwaige Distributionskosten sind kein Entgelt für die Güterproduktion und sollen daher im Folgenden insoweit außer Betracht bleiben.

7 Sicherlich ist es möglich, auch immaterielle Entlohnungen wie Anerkennung und Prestigegewinn als Engelte im weiteren Sinne zu verstehen, vgl. etwa Ghosh (1998). Ein solches Verständnis wirtschaftlicher Handlungen würde aber jedes menschliche Handeln, soweit es der Erreichung bestimmter Ziele dient, erfassen; es würde auf diese Weise konturenlos werden und eher menschliches Verhalten allgemein als spezifisch die Mechanismen beschreiben, die der Güterverteilung dienen, vgl. auch (Koenig und Neumann 2004*b*, S. 130, Fn. 184).

8 Aus diesem Grund erweist sich auch das EG-Wettbewerbsrecht (Art. 81 ff. EG-Vertrag) als wenig geeignet zur Anwendung auf Sachverhalte mit Bezug zu quelloffener Software, vgl. Koenig und Neumann (2003).

9 Bei entgeltlicher quelloffener Software bestehen keine grundsätzlichen Unterschiede zu anderen Handelsgütern. Auch ein quelloffenes Handelsgut ist ein Handelsgut.

10 Siehe auch Wiebe (2003, S. 163).

Wiebe (2003, S. 163)). Nicht ihr Angebot, aber ihre Produktion ist in diesen Fällen folglich eine typische (markt-) wirtschaftliche Tätigkeit. Auch die Distribution freier quelloffener Software hat (markt-) wirtschaftliches Potential bewiesen,[11] wie namentlich das Aufkommen zahlreicher Linux-Distributionen (und -Distributoren) gezeigt hat. Ähnliche Effekte sind auf den dem eigentlichen Angebot nachgelagerten Ebenen zu verzeichnen: Mit zunehmender Verbreitung und Nutzung freier quelloffener Software sind zugleich der Bedarf und das Angebot an entsprechenden – in erheblichen Teilen durchaus entgeltlichen – Servicedienstleistungen gewachsen (Koenig und Neumann 2004*b*, S. 130; Wiebe 2003, S. 163), wozu auch entsprechende Fachliteratur (Zeitschriften und Bücher) zählt. Hinzu kommen schließlich die effizienz- und innovationserhöhenden Auswirkungen, die von freier quelloffener Software ausgehen können: Durch die Existenz eines kaum mehr überschaubaren Angebots an kostenlosen Programmen besteht die Möglichkeit, ohne allzu große Transaktionskosten verschiedene Softwarelösungen für einen bestimmten Bedarf umfassend zu testen, zu vergleichen und gegebenenfalls auch parallel zu nutzen. Auch Pfadabhängigkeiten bei der Nutzung von proprietären Lösungen können durch freie quelloffene Software vermieden werden. Zugleich führt der Austausch von Programmcode bei quelloffener Software zu einer Aktivierung von Netzeffekten bei der Softwareentwicklung. Die dadurch erzielbaren Innovationseffekte erlauben ihrerseits die Schaffung neuer Programmlösungen (Blind et al. 2001, S. III) und fördern auf diese Weise die Innovationsdynamik (Blind et al. 2001, S. VII) und damit letzten Endes auch die Güterproduktion.

Quelloffene Software ist somit eine Herausforderung für das EG-Primärrecht, wobei dieser Befund zwei unterschiedliche Seiten hat: Auf der einen Seite passt freie quelloffene Software nicht in das dem ökonomischen Ansatz des EG-Vertrages zugrunde liegende marktwirtschaftliche Konzept, das auf der Prämisse unmittelbarer Austauschverhältnisse beruht. Stattdessen kann sie derartige Austauschverhältnisse sogar in solch einer Weise beeinflussen, dass der Güteraustausch auf den betroffenen Märkten zurückgeht. Freie quelloffene Software ist insoweit „Wettbewerber des Wettbewerbs". Auf der anderen Seite kann sie den Wettbewerb auf solchen Ebenen fördern, die dem eigentlichen Produktangebot vor- und nachgelagert sind, insbesondere auf Dienstleistungsmärkten.[12] überdies kann sie die gesamtwirtschaftliche Effizienz und Innovation fördern, was ebenfalls wettbewerbsstimulierend wirkt. Die Analyse des EG-Primärrechts ergibt damit ein disparates Bild. Hieraus erklärt sich, dass die Gemeinschaft bislang noch keine übergeordnete Strategie mit Blick auf (freie) quelloffene Software entwickelt hat (bzw. überhaupt entwickeln konnte), wie es sie

11 Vgl. auch Schulz (2004, S. 573–574).

12 Auch in der Stellungnahme des Wirtschafts- und Sozialausschusses zu der „Mitteilung der Kommission an den Rat und das Europäische Parlament, den Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen: Sicherheit der Netze und Informationen: Vorschlag für einen europäischen Politikansatz" (2002/C 48/07), ABl. EG 2002 C 48, 33, 35, wird konstatiert, dass „sich ein wichtiger Wirtschaftssektor von Diensten für Unternehmen" um das Konzept freier quelloffener Software „entwickelt (habe), der von einigen Grossunternehmen des Informatiksektors getragen wird." Der Ausschuss sieht in freier quelloffener Software „eine Form gesunder Konkurrenz zu den monopolistischen Tendenzen des Softwaremarktes sowie des sich ständig weiterentwickelnden Marktes an Netzdiensten".

in anderen Bereichen – etwa der Telekommunikation oder des Eisenbahnwesens – gibt. Quelloffene Software spielt daher in der Realität der EG primär auf einzelnen Mikroebenen eine Rolle.

3. Die erste Mikroebene: quelloffene Software als Beitrag zur Sicherstellung von Interoperabilität

Ein Wirtschaftsbereich, der in der Politik der Gemeinschaft in den letzten fnfzehn Jahren eine zunehmend prominente Rolle eingenommen hat, ist die Branche der Informations- und (Tele-) Kommunikationstechnologien. Seit 1990 hat die EG konsequent und zielstrebig die Öffnung des zuvor in fast allen europäischen Ländern staatsmonopolistisch organisierten Telekommunikationssektors betrieben.[13] Bis zum 1. Januar 1998 mussten die Mitgliedstaaten (grundsätzlich) den Betrieb von Telekommunikationsnetzen und das Angebot von Telekommunikationsdiensten vollständig liberalisiert haben, so dass der Telekommunikationssektor auf allen Ebenen dem Wettbewerb geöffnet war. Dieser Wettbewerb wurde zugleich durch ein umfassendes System aus Netzzugangsansprchen und flankierender Regulierung gefördert, das nach einer grundlegenden Reform im Jahr 2002 in konsolidierter und ausdifferenzierter Form,[14] auf bislang nicht absehbare Zeit fortbestehen wird. Diese Liberalisierung der Telekommunikationsmärkte traf auf eine technische Entwicklung von radikaler Dynamik, in deren Folge Informations- und Kommunikationstechnologien, vor allem auch auf Grundlage des Internets, so gut wie alle Lebensbereiche erfasst und durchdrungen haben. Die dadurch entstandenen Märkte sind praktisch ausnahmslos durch Netzcharakteristika geprägt. Netzeffekte – also insbesondere der (grundsätzliche) Nutzenzuwachs, der für alle Teilnehmer aus dem Anschluss eines zusätzlichen Teilnehmers folgt – führen aber grundsätzlich zu einem natürlichen Wettbewerbsvorteil großer Netze (Koenig et al. 2004, S. 42). Diese ökonomische Besonderheit birgt die wettbewerbspolitische Gefahr, dass es zur Bildung enger Oligopole oder sogar zur Monopolstellung eines einzelnen Netzes und damit zu nicht wettbewerblich kontrollierten Machtpositionen der betreffenden Anbieter kommt. Doch auch wenn eine solche Entwicklung vermieden werden kann, begründen Netzeffekte jedenfalls potentielle Pfadabhängigkeiten, die zu gesamtwirtschaftlich nicht wünschenswerten Ergebnissen führen können (Koenig und Neumann 2004*b*, S. 116). Ein zentrales Ziel, das die Gemeinschaft im Bereich der Informations- und Kommunikationstechnologien verfolgt, ist daher die Sicherstellung von Interoperabilität.[15] Wenn (Netz-) Systeme mit anderen (Netz-) Systemen zusammenarbeiten können, sind Netzeffekte erzielbar, ohne dass das Gesamtsystem unter der Kontrolle nur eines Betreibers stehen muss.

13 Ausführlich hierzu und mit umfangreichen Nachweisen zu den jeweiligen Maßnahmen (Koenig et al. 2004, S. 57 ff.).

14 Siehe hierzu Koenig et al. (2004, S. 73 ff.).

15 Siehe ausdrücklich zum Stellenwert von Interoperabilitätsregulierung die Mitteilung der Kommission über Hemmnisse für den breiten Zugang zu neuen Diensten und Anwendungen der Informationsgesellschaft durch offene Plattformen beim digitalen Fernsehen und beim Mobilfunk der dritten Generation, KOM (2003) 410, S. 8.

Zugleich kann es in Teilsystemen zu technischer Innovation kommen, ohne dass dabei auf die Vorteile eines Rückgriffs auf die installierte Basis der anderen Teilsysteme verzichtet werden msste.[16]

Dabei hat die EG die besondere Eignung von quelloffener Software zur Sicherstellung von Interoperabilität erkannt. Zwar erfordert Interoperabilität im Wesentlichen nur die Offenheit von *Schnittstellenspezifikationen* und nicht auch der weiteren Netzkomponenten (Blind et al. 2001, S. III). Sind jedoch die Komponenten selbst offen, sind es zwangsläufig auch die Schnittstellen.[17] Gerade angesichts der Erfahrungen mit der nicht immer ausgeprägten Neigung von Softwareherstellern, die Schnittstellen ihrer Programme vollständig und auf wettbewerbsneutrale Weise offen zu legen,[18] kommt der Verbreitung quelloffener Software – und ihrer Förderung – insoweit ein besonderer Wert zu. Da die reaktive Herstellung der Offenheit von Schnittstelleninformationen mit den Mitteln des allgemeinen Wettbewerbsrechts zeitintensiv und auch darüber hinaus mit erheblichen praktischen Schwächen belastet ist (Koenig et al. 2003, S. 404 ff.), gilt dies gerade in den dynamischen Softwaremärkten und benachbarten Märkten der Information und Kommunikation in noch gesteigertem Maß. Die EG erkennt diesen Nutzen quelloffener Software an und operationalisiert sie in geeigneten Teilbereichen zur Erreichung des generellen Interoperabilitätsziels. Da die Verfügbarkeit quelloffener Software aber von dem Eigenengagement ihrer Hersteller abhängig ist und dieses zumindest in der Mehrzahl der Fälle nicht von wirtschaftlichen (oder gar rechtlichen) Überlegungen geleitet ist, sind die direkten Steuerungsmöglichkeiten der Gemeinschaft insoweit sehr reduziert. Sie muss sich daher im Wesentlichen auf eine Förderung quelloffener Software als Bestandteil ihrer Interoperabilitätsstrategie beschränken.

So begründete die Kommission der Europäischen Gemeinschaften in ihrer Mitteilung zur Organisation und Verwaltung des Internets schon im Jahr 2000 ihre Absicht „sicherzustellen, dass die bestehende Neutralität der Internet-Spezifikationen in Bezug auf Betriebssysteme und Hardware-Plattformen beibehalten und weiterentwickelt wird", primär mit dem „zunehmenden Interesse [. . .] der Nutzer an offener Software".[19] Die Gemeinschaft setzt aber vor allem mit ihrer Förderungspolitik explizit positive Anreize, verstärkt quelloffene Software zu entwickeln und einzusetzen. Diese Anreizsetzungen sind in einem breiteren Kontext der von der Kommission vorbereiteten und vom Europäischen Rat im Jahr 2000 beschlossenen Strategie für einen beschleunigten Übergang zu einer wettbewerbsfähigen und dynamischen wissensgestützten Wirtschaft, dem so genannten „Aktionsplan *e*Europe", zu sehen. Die

16 Siehe zum Ganzen auch ausführlich Koenig et al. (2003, S. 410 ff.).

17 Dieser Erst-recht-Zusammenhang wird auch in der Mitteilung der Kommission an den Rat, das Europäische Parlament, den Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen "Sicherheit der Netze und Informationen: Vorschlag für einen europäischen Politikansatz", KOM (2001) 298 endgültig, S. 25, betont.

18 Siehe hierzu die Microsoft-Entscheidung der Kommission, WuW 2004, 673, 676 ff., und dazu Koenig und Neumann (2004*a*, S. 555, 559) m. w. Nachw, sowie allgemein zur Schnittstellproblematik Koenig und Neumann (2004*b*, S. 132 f.) ebenfalls m. w. Nachw.

19 Mitteilung der Kommission an den Rat und das Europäische Parlament „Organisation und Verwaltung des Internets – Internationale und europäische Grundsatzfragen 1998–2000", KOM (2000) 202, S. 12.

Förderung von quelloffener Software war – beschränkt auf einzelne Sachgebiete, namentlich den Bereich sicherer Netze und intelligenter Chipkarten sowie die Initiativen „Europas Jugend ins Digitalzeitalter" und „Regierung am Netz – elektronischer Zugang zu öffentlichen Dienstleistungen", – bereits in dem ursprünglichen Aktionsplan selbst angelegt und als Aufgabe gerade auch der Kommission und der Mitgliedstaaten definiert.[20] Dieser Konzentration auf die drei – hier der Einfachheit halber im Einklang mit einer fragwürdigen Praxis über englische Kunstbegriffe bezeichneten[21] – Schwerpunktbereiche *e*Security, *e*Learning und *e*Government entspricht die nachfolgende Praxis der Gemeinschaftsorgane:

Bereits im Jahr 2001 forderte der Rat im Zusammenhang mit der Integration von Informations- und Kommunikationstechnologien in die Bildungs- und Ausbildungssysteme (*e*Learning) die Kommission auf, „die Entwicklung von ... 'Open-Source-Software' zu fördern".[22] Im selben Jahr empfahl die Kommission inInteroperabilität ihrer Mitteilung über die Sicherheit der Netze und Informationen überdies die Verwendung quelloffener Software zur Sicherstellung der Interoperabilität von sicherheitsfördernden Lösungen – etwa in Bezug auf elektronische Signaturen –, die überdies „zur schnelleren Fehlerkorrektur sowie zu größerer Transparenz beitragen" könne.[23] Und erst im Frühjahr 2004 berücksichtigten das Europäische Parlament und der Rat in ihrem Programm für den Zeitraum 2005–2009 zur interoperablen Erbringung europaweiter elektronischer Behördendienste („eGovernment-Dienste") für europäische öffentliche Verwaltungen, die Organe der Gemeinschaft und andere Stellen sowie europäische Unternehmen und Bürger[24] ausdrücklich quelloffene Software bei der Festlegung der förderungswürdigen Maßnahmen. Die Bereitstellung und Wartung von „Werkzeuge(n), die auf quelloffener Software basieren", werden dabei als Infrastrukturdienst identifiziert.[25] Als solcher kommen sie für die Durchführung als (von der Gemeinschaft mitfinanzierte) horizontale Maßnahmen im Sinne des Programms in Betracht.[26] Darüber hinaus wird auch die Förderung der Verbreitung bewährter Verfahren bei der Nutzung von quelloffener Software durch öffentliche Verwaltungen als flankierende Maßnahme ebenfalls in den Anwendungsbereich des Förderungsprogramms einbezogen.[27]

Der *e*Europe-Aktionsplan wurde im Jahr 2002 zum „Aktionsplan *e*Europe 2005"[28]

20 Kommission der Europäischen Gemeinschaften, *e*Europe 2002: Eine Informationsgesellschaft für alle – Aktionsplan vorbereitet von Rat und Europäischer Kommission zur Vorlage auf der Tagung des Europäischen Rates am 19./20. Juni 2000 in Feira, 2000, S. 11, 13 und 22.

21 Siehe generell zu fremdsprachigen Einflüssen auf die deutsche Sprache Stickel (2001, S. 159).

22 Entschließung des Rates vom 13. Juli 2001 zum *e*-Learning (2001/C 204/02), ABl. EG 2001 C 204, 3, Entschließungspunkt 10 viii).

23 Kommission, KOM (2002) 202, S. 25.

24 Beschluss 2004/387/EG des Europäischen Parlaments und des Rates vom 21. April 2004 über die interoperable Erbringung europaweiter elektronischer Behördendienste (eGovernment-Dienste) für öffentliche Verwaltungen, Unternehmen und Bürger (IDABC) – in der berichtigten Fassung –, ABl. EG 2004 L 181, 25.

25 Anhang II B. lit. r des Beschlusses 2004/387/EG.

26 Vgl. Art. 5, 6 und 10 des Beschlusses 2004/387/EG.

27 Anhang II C. 3. lit. c des Beschlusses 2004/387/EG.

28 Kommission der Europäischen Gemeinschaften, Mitteilung der Kommission an das Europäische Parla-

hin fortentwickelt. Dabei wurde das Ziel, quelloffene Software – gerade in den genannten drei Bereichen *e*Security, *e*Learning und *e*Government – zu fördern, fortgeschrieben[29] und nunmehr zum Teil deutlich in den übergeordneten funktionalen Zusammenhang der Sicherstellung von Interoperabilität gerückt.[30] Weitere Maßnahmen zur Unterstützung und Ergänzung des Aktionsplans *e*Europe 2005 wurden daraufhin auf Ebene der Forschungsförderung unter dem Themenbereich „Technologien für die Informationsgesellschaft" (TIS)[31] vorgesehen, die ihrerseits auch ausdrücklich quelloffene Software berücksichtigten. Erneut wurde die Förderung „von Software mit frei zugänglichem Quellcode" als wichtige Maßnahme definiert, „wenn dies erforderlich ist, um die Interoperabilität der Lösungen sicherzustellen und Innovationen voranzubringen".[32] Für elektronische Behördendienste wurde hier sogar eine generelle Priorität für den Einsatz quelloffener Software definiert.[33] Auch für den Bereich elektronischer Gesundheitsdienste (*e*Health) wurde insoweit nunmehr grundsätzlich eine Förderung der Verwendung quelloffener Software in Aussicht gestellt.[34] Außerdem wurde erneut mehrfach auf den Interoperabilitätsnutzen verwiesen, der durch den Einsatz quelloffener Software – etwa im Bereich der Softwareentwicklung und der Forschungstätigkeit – erzielt werden kann.[35]

4. Die zweite Mikroebene: Softwarepatente als Bedrohung für quelloffene Software

Auf einer zweiten Mikroebene, auf der quelloffener Software in der Politik der Europäischen Gemeinschaft besondere Bedeutung zukommt, geht es weniger um die Frage, ob und wofür quelloffene Software nützlich ist. Stattdessen geht es bei der nunmehr bereits seit einigen Jahren sehr engagiert geführten Debatte über Softwarepatente – und ihre explizite normative Harmonisierung durch die Gemeinschaft – um die Frage, welche Auswirkungen die eventuell an nicht sonderlich hohe Voraus-

ment, den Rat, den Wirtschafts- und Sozialausschuss und den Ausschuss der Regionen „*e*Europe 2005: Eine Informationsgesellschaft für alle – Aktionsplan zur Vorlage im Hinblick auf den Europäischen Rat von Sevilla am 21./22. Juni 2002", KOM (2002) 263 endg.

29 Kommission der Europäischen Gemeinschaften, Aktionsplan *e*Europe 2005, KOM (2002) 263 endg., S. 12 und 18. Lediglich im Bereich des *e*Learnings fand quelloffene Software keine ausdrückliche Erwähnung mehr auf Ebene des Aktionsplans selbst.

30 Vgl. insbesondere Kommission der Europäischen Gemeinschaften, Aktionsplan *e*Europe 2005, KOM (2002) 263 endg., S. 12 („Bis Ende 2003 wird die Kommission einen abgestimmten Rahmen für die Interoperabilität bekannt geben, der die Bereitstellung europaweiter elektronischer Behördendienste für Bürger und Unternehmen unterstützen soll. [. . .] Grundlage werden offene Normen sein, und die Verwendung von Software mit frei zugänglichem Quellcode wird unterstützt.").

31 Kommission der Europäischen Gemeinschaften, Technologien für die Informationsgesellschaft: Ein vorrangiger Themenbereich für Forschung und Entwicklung im Rahmen des Spezifischen Programms „Integration und Stärkung des Europäischen Forschungsraums" des 6. Rahmenprogramms der Gemeinschaft – Arbeitsprogramm 2003 – 2004.

32 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 7.

33 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 20 („Sie sollten . . . soweit wie möglich Softwarelösungen mit frei zugänglichem Quellcode wählen.").

34 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 22.

35 Kommission der Europäischen Gemeinschaften, TIS-Arbeitsprogramm 2003 – 2004, S. 26 und 38.

setzungen geknüpfte Möglichkeit eines Patentschutzes für Rechnerprogramme auf quelloffene Software hätte und ob sich diese Konsequenzen angesichts der Vorteile von Softwarepatenten rechtfertigen lassen. Diskutiert wird die Frage anlässlich des Gesetzgebungsverfahrens zu einer Richtlinie über die Patentierbarkeit computerimplementierter Erfindungen. Mit dieser Richtlinie soll vor allem die Anwendung des Patentrechts bei der Erteilung von Softwarepatenten in Rechtsprechung und Praxis der Mitgliedstaaten vereinheitlicht werden, um so Hemmnisse für den Binnenmarkt abzubauen. Schon bevor die Kommission im Jahr 2002 ihren initialen Richtlinienvorschlag[36] vorlegte, hatten sich Entwickler und Anwender von quelloffener Software als Hauptgegner einer ausdrücklichen Regelung (bzw. grundsätzlichen Bestätigung) von Softwarepatenten zu erkennen gegeben.[37]

Die Diskussion um Softwarepatente führt im Kern auf den grundlegenden Widerspruch gewerblicher Schutzrechte zurück, die der Sache nach hoheitlich gesicherte Monopolrechte sind, von denen eine wettbewerbsfördernde Wirkung ausgehen soll. Dahinter steht die Überlegung, dass die Aussicht auf Monopolgewinne Unternehmer und Erfinder zu Innovationen antreibt.[38] Zum Teil wird sogar noch weiter gehend zwischen „Gütern", „Produktion" und „Innovation" als drei Wettbewerbsebenen unterschieden und davon ausgegangen, dass Wettbewerb auf einer Ebene eine Beschränkung des Zugangs auf der darunter liegenden Ebene erforderlich macht.[39] Danach setzt Wettbewerb auf Ebene der Produktion Eigentum an Gütern, also den Ausschluss des freien Zugangs zu bestehenden Gütern voraus. Und Wettbewerb auf der Innovationsebene würde wiederum die Unterdrückung von Wettbewerb auf der Produktionsebene erfordern, was gerade durch gewerbliche Schutzrechte wie Patente oder das Urheberrecht erreicht würde.[40] Auch wenn die Hypothese, dass die Aussicht auf Marktmacht Treiber für Innovationen ist, bislang empirisch nicht erhärtet wurde, ist sie plausibel und in einem gewissen Umfang sicherlich auch zutreffend (Koenig et al. 2002, S. 28). Auch bei den Softwarepatenten ist die eigentliche Frage daher diejenige nach diesem Umfang.

Indem Patente funktionale Aspekte schützen, erfasst ein Softwarepatent unabhängig von der konkreten Implementierung jedes Programm, das einen Rechner zu einem patentgeschützten Verhalten veranlasst (Gehring und Lutterbeck 2003, S. 6; Wiebe 2003, S. 163). Da Softwarepatente einerseits verhältnismäßig preiswert zu produzieren sind und andererseits bei einem Programmablauf regelmäßig eine Vielzahl von – potentiell patentgeschützten – Funktionen erfüllt werden, hat die Zulassung von

36 Kommission der Europäischen Gemeinschaften, Vorschlag für eine Richtlinie des Europäischen Parlaments und des Rates über die Patentierbarkeit computerimplementierter Erfindungen, KOM (2002) 92 endgültig.

37 Vgl. Kommission der Europäischen Gemeinschaften, KOM (2002) 92 endgültig, S. 3.

38 Daneben wird oftmals darauf verwiesen, dass die Monopolgewinne zur (Re-) Finanzierung kostspieliger Innovationen auch benötigt werden. Dieses Argument ist offensichtlich nicht generell gültig, sondern auf Branchen mit entsprechenden Kostenstrukturen beschränkt. Für die Pharmaindustrie dürfte es daher eher Gültigkeit beanspruchen als – zumindest bei genereller Heranziehung – für die Softwarebranche.

39 von Weizsäcker (1980, passim)

40 Siehe hierzu auch Koenig et al. (2002, S. 54).

Softwarepatenten zur Folge, dass sich Softwareproduzenten einem „Patentdickicht“[41] gegenübersehen und Softwareentwicklung mit (zumindest potentiell) hohen Lizenz- bzw. Ausweichkosten behaftet wird.[42] Für die Anbieter kommerzieller Software müssen insoweit wechselseitige Lizenzen (*Cross Licensing*) und die Einrichtung so genannter Technologiepools[43] praktische Problemlösungsstrategien darstellen (Gehring und Lutterbeck 2003, S. 16 f.) oder letzten Endes eine Abwälzung der Kosten auf den Käufer in Betracht kommen. Diese Möglichkeiten haben die Entwickler (freier) quelloffener Software hingegen in der Regel nicht(Gehring und Lutterbeck 2003, S. 18). Softwarepatente können für quelloffene Software daher tendenziell existenzbedrohende Auswirkungen haben.[44]

Bislang ist noch nicht absehbar, wie dieser grundlegende Interessenkonflikt auf europäischer Ebene letzten Endes entschieden werden wird. Das Europäische Parlament hatte in der ersten Lesung des Richtlinienentwurfs nicht nur eine besondere Beobachtung der Auswirkungen computerimplementierter Erfindungen auf die Open-Source-Bewegung vorgeschrieben. Viel weitgehender hatte es vor allem die Anforderungen an patentwürdige Software gegenüber dem ursprünglichen Vorschlag der Kommission erheblich angehoben und insbesondere eine Ausnahmeregelung für die Verwendung patentierter Technologien zum Zwecke der Interoperabilität vorgesehen.[45] Der Rat legte sich hingegen in einer am 18. Mai 2004 erzielten politischen Einigung auf eine wieder stärker dem Kommissionsvorschlag angenäherte Fassung fest, in der die Einschränkungen, die das Europäische Parlament vorgesehen hatte, wieder weitgehend zurückgenommen waren.[46] Von mitgliedstaatlicher Ebene werden jedoch seitdem zunehmend Bedenken gegenüber dem im Rat gefundenen Kompromiss laut – nach einem Beschluss des niederländischen Parlaments, in welchem die niederländische Regierung aufgefordert wird, der Einigung im Rat die Unterstützung zu entziehen, liegen nunmehr auch ähnlich ausgerichtete Entschließugsanträge der

41 Zum Begriff siehe Shapiro (2000, passim.).

42 Siehe zum Ganzen etwa Gehring und Lutterbeck (2003, S. 6 ff.), m. w. Nachw.

43 Siehe zu diesem Modell Kommission der Europäischen Gemeinschaften, Leitlinien zur Anwendung von Artikel 81 EG-Vertrag auf Technologietransfer-Vereinbarungen, ABl. EG 2004 C 101, 2, Rn. 210 ff., und dazu (Koenig und Neumann 2004*a*, S. 555,558 f.).

44 Dementsprechend wird eine Ausweitung der Patentierbarkeit von Software durch Entwickler quelloffener Software generell negativ bewertet und auch von anderen (kommerziellen) Softwareproduzenten Einschränkungen bei der Entwicklung (freier) quelloffener Software als Folge einer (weiteren) Ausweitung der Patentierbarkeit erwartet Blind et al. (2001, S. V f.). Keine ernsthafte Gefährdung für quelloffene Software erwartet hingegen Wiebe (2003, S. 163), der jedoch von „weiteren erheblichen Hürden für eine Patentierung“ ausgeht, die in der Realität zurzeit freilich zusehends abgebaut werden.

45 Bericht über den Vorschlag für eine Richtlinie des Europäischen Parlaments und des Rates über die Patentierbarkeit computerimplementierter Erfindungen, A5-0238/2003 endgültig; Standpunkt des Europäischen Parlaments festgelegt in erster Lesung am 24. September 2003 im Hinblick auf den Erlass der Richtlinie 2003/.../EG des Europäischen Parlaments und des Rates über die Patentierbarkeit computerimplementierter Erfindungen, ABl. EG 2004 C 77 E, 230. Siehe hierzu auch Hemler (2004), EuZW 2004, 257, unter Hinweis auf die Notwendigkeit einer Vereinheitlichung des Softwareschutzes in der Rechtspraxis der USA und der EG.

46 Rat der Europäischen Union, Politische Einigung über den gemeinsamen Standpunkt des Rates, Dok. 9713/04.

Fraktionen des Deutschen Bundestages vor.[47] Es ist somit noch nicht einmal gesichert, dass auf die politische Einigung die (momentan für Ende November 2004 vorgesehene) formelle Verabschiedung eines gleichlautenden gemeinsamen Standpunktes folgt. Selbst wenn der Rat aber bei seiner Linie bleibt, ist derzeit nicht prognostizierbar, wie sich das Europäische Parlament in der dann anstehenden zweiten Lesung zu dieser Richtlinienentwurfsfassung positionieren wird. Dies gilt in besonderem Maß, da das Parlament aufgrund der zwischenzeitlich erfolgten Neuwahlen neu zusammengesetzt ist und somit noch nicht einmal von einer politischen Kontinuität ausgegangen werden kann.

5. Fazit

Die EG hat noch keine einheitliche Haltung zu quelloffener Software gefunden. Die Idee des (regelmäßig) unentgeltlichen Angebots von Gütern lässt sich nicht ohne Friktionen in den Grundansatz einer auf den unmittelbaren Leistungsaustausch gerichteten Wirtschaftsgemeinschaft integrieren, um die es sich bei der EG im Kern nach wie vor handelt. Eine Analyse der betroffenen Mikroebenen spiegelt diesen grundsätzlichen Zielkonflikt, der auf der Makroebene des Primärrechts besteht, deutlich wider: Einerseits erkennt die Gemeinschaft die Schnittstellenfunktion von quelloffener Software als Plattform zur Herstellung von Interoperabilität mit all ihren wettbewerbsfördernden Wirkungen insbesondere auf benachbarten Märkten an. Andererseits nehmen zumindest die Kommission und der Rat zur Förderung des Wettbewerbs auf den Softwaremärkten eine durchaus (zumindest potentiell) existentielle Bedrohung quelloffener Software in Kauf, indem sie trotz der zusehends ausufernden Anwendungspraxis Softwarepatente ausdrücklich normativ anerkennen wollen. Zugleich zeigt der Streit um die Softwarepatente die Schwierigkeiten auf, die wettbewerbs- und wohlfahrtsfördernden Wirkungen quelloffener Software in das überkommene System der europäischen Wettbewerbspolitik einzupassen. Die auf Mikroebene festgestellten Widersprüche verdeutlichen überdies auch, dass die Gemeinschaft zumindest zurzeit noch keine übergeordnete Strategie in Bezug auf quelloffene Software verfolgt. Das ist Risiko und Chance zugleich: Das Risiko liegt darin, dass auf den einzelnen Mikroebenen nicht aufeinander abgestimmte Strategien verfolgt werden und es auf diese Weise zu Widersprüchlichkeiten und Fehlentwicklungen kommt. Die Chance besteht darin, dass die Mitgliedstaaten noch in der Lage sind, einen künftigen übergeordneten Ansatz der Gemeinschaft mit Blick auf quelloffene Software proaktiv mitzugestalten. Sie sollten diese Chance nutzen.

47 Vgl. die Anträge der Fraktion der FDP, BT-Drs. 15/3240, der Fraktion der CDU/CSU, BT–Drs. 15/3941, sowie der Fraktionen der SPD und von Bündnis 90/Die Grünen, BT-Drs. 15/4034.

Literaturverzeichnis

Blind, K., Edler, J., Nack, R. und Straus, J. (2001), 'Mikro- und makroökonomische Implikationen der Patentierbarkeit von Softwareinnovationen: Geistige Eigentumsrechte in der Informationstechnologie im Spannungsfeld von Wettbewerb und Innovation (Kurzfassung)', Karlsruhe.

Calliess, C. und Ruffert, M. (Hrsg.) (2002), *Kommentar zu EU-Vertrag und EG-Vertrag*, 2. Aufl., Hermann Luchterhand Verlag, Neuwied/Kriftel.

Gehring, R. A. und Lutterbeck, B. (2003), 'Software-Patente im Spiegel von Softwareentwicklung und Open Source Software', http://ig.cs.tu-berlin.de/ma/rg/ap/2003-x/GehringLutterbeck-SWPat-092003.pdf/file/.

Ghosh, R. A. (1998), 'Cooking pot markets: an economic model for the free trade in free goods and services on the Internet', *First Monday* **3**(3).

Hemler, T. (2004), 'Softwarepatente – quo vadis?', *Europäische Zeitschrift für Wirtschaftsrecht* S. 257.

Koenig, C. und Haratsch, A. (2003), *Europarecht*, Mohr Lehrbuch, 4. Aufl., J. C. B. Mohr (Paul Siebeck), Tübingen.

Koenig, C., Loetz, S. und Neumann, A. (2003), Innovation im Spannungsverhältnis von Markt und Regulierung, *in* D. Klumpp, H. Kubicek und A. Roßnagel (Hrsg.), 'Next generation information society?', Talheimer Verlag, Mössingen-Talheim, S. 403–412.

Koenig, C., Loetz, S. und Neumann, A. (2004), *Telekommunikationsrecht*, Betriebs-Berater Studium, UTB/Verlag Recht und Wirtschaft, Heidelberg.

Koenig, C. und Neumann, A. (2003), 'Standardisierung und EG-Wettbewerbsrecht – ist bei vertrauenswürdigen Systemumgebungen wettbewerbspolitisches Misstrauen angebracht?', *Wirtschaft und Wettbewerb* S. 1138–1152.

Koenig, C. und Neumann, A. (2004*a*), 'Neue wettbewerbspolitische und -rechtliche Entwicklungen zum „Trusted Computing"', *Datenschutz und Datensicherheit* S. 555–560.

Koenig, C. und Neumann, A. (2004*b*), Wettbewerbsrechtliche Aspekte vertrauenswürdiger Systemumgebungen, *in* C. Koenig, A. Neumann und T. Katzschmann (Hrsg.), 'Trusted Computing', Schriftenreihe Kommunikation & Recht, Verlag Recht und Wirtschaft, Heidelberg, S. 100–140.

Koenig, C., Vogelsang, I., Kühling, J., Loetz, S. und Neumann, A. (2002), *Funktionsfähiger Wettbewerb auf den Telekommunikationsmärkten*, Schriftenreihe Kommunikation & Recht, Verlag Recht und Wirtschaft, Heidelberg.

Loetz, S. und Neumann, A. (2003), 'The Scope of Sector-specific Regulation in the European Regulatory Framework for Electronic Communications', *German Law Journal* S. 1307–1334.

Schulz, C. (2004), 'Open Source Software vor Gericht', *Multimedia und Recht* S. 573–574.

Shapiro, C. (2000), 'Navigating the Patent Thicket: Cross Licenses, Patent Pools, and Standard-Setting', Working Paper No. CPC00-11, Berkeley.

Stickel, G. (2001), Das „Fremdwort" hat ausgedient, *in* A. Koch und A. Neumann (Hrsg.), 'Ut desint vires, tamen est laudanda voluntas – Festschrift für Christian Celsen zum Bestehen des ersten juristischen Staatsexamens', Edition Octopus, Verlagshaus Monsenstein und Vannerdat, Münster, S. 159–170.

Wiebe, A. (2003), 'Open Source Software – eine erfolgreiche Alternative', *Telekommunikations- & Medienrecht* S. 163.

von Weizsäcker, C. C. (1980), *Barriers to Entry: A Theoretical Treatment*, Springer-Verlag, Berlin/Heidelberg/New York.

Der Kampf gegen Softwarepatente – Open Source im Auge des Sturms

STEFAN KREMPL

Am Brüsseler Richtlinienprojekt über die „Patentierbarkeit computerimplementierter Erfindungen“ scheiden sich die Geister: Konzerne in der Computer- und Telekommunikationsindustrie erhoffen sich davon eine Harmonisierung der Rechtssituation innerhalb der EU und eine Sanktionierung der bereits recht weitgehenden Praxis des Europäischen Patentamtes. Überraschend starke Lobbygruppen aus dem Mittelstand warnen dagegen vor einer Flut an „Trivialpatenten“ nach US-Muster und schwerwiegenden Auswirkungen auf den Wettbewerb allgemein sowie auf die Entwicklung freier Software im Besonderen. Das Gesetzgebungsverfahren gestaltet sich daher als rechtlicher Krimi mit Klassenkämpfen, Ränkespielen und zahlreichen Verzögerungen. Auch im dritten Jahr nach der Veröffentlichung des ersten Richtlinienvorschlags zeichnet sich noch kein Kompromiss zwischen Kommission, Rat und Parlament ab. Der Beitrag gibt einen Überblick über die wichtigsten Frontlinien.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Tragen die Juristen Open-Source-Software zu Grabe? – Die GNU GPL vor Gericht

THOMAS EBINGER

(CC-Lizenz siehe Seite 463)

Das Recht bedroht den zum Riesen heranwachsenden Zwerg Open Source. Es bestehen im Wesentlichen zwei Gefahren: (a) die fehlende Durchsetzbarkeit von Open-Source-Lizenzen in Deutschland und (b) die Verletzung von softwarebezogenen Patenten durch Open-Source-Software (OSS). Beide Bedrohungen könnten das etablierte Softwareentwicklungsmodell in den Grundfesten erschüttern und das Ende von OSS einläuten. Die GNU General Public License (GPL) ist die bekannteste Open-Source-Lizenz. Deren Besonderheiten – sie ist nach US-amerikanischem Recht erstellt, liegt offiziell nur in englischer Sprache vor und die hierunter entwickelte Software wird international im Internet entwickelt – bergen gewisse rechtliche Schwierigkeiten bei ihrer Anwendung in Deutschland. Mit dem Urteil des Landgerichts München I vom 19.05.2004 liegt nun eine erste Entscheidung eines deutschen Gerichts zur GPL vor. Die Richter befassen sich u. a. mit folgenden Fragen: Kann die Nutzung einer GPL-Software untersagt werden, sofern gegen bestimmte Klauseln der GPL verstoßen wird? Wer kann Rechte aus OSS (gerichtlich) geltend machen? Unter welchen Voraussetzungen gilt die GPL? Gilt die Lizenz in der deutschen Rechtsordnung trotz des englischen Lizenztextes? Das Mehr an Rechtssicherheit durch diese Gerichtsentscheidung wird kontrastiert durch die ungelöste Bedrohung von OSS durch Softwarepatente.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Kapitel 5

Gesellschaft

Open Source – Zwischen Geschichte und Zukunft

SEBASTIAN ZIEBELL

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Anarchie und Quellcode – Was hat die freie Software-Bewegung mit Anarchismus zu tun?

CHRISTIAN IMHORST

(FDL 1.2 siehe Seite 466)

Wie anarchistisch ist die Hackerethik? Warum bezeichnet sich Richard Stallman, eine der herausragendsten Figuren der freien Software-Bewegung, selbst als Anarchist? Wir dürfen uns die Anarchisten der freien Software nicht klischeehaft als Chaoten mit zerzausten Haaren, irrem Blick und Armen voller Bomben vorstellen. Ganz im Gegenteil: Sie fordern eine neue Ordnung des „geistigen Eigentums" im Sinn der Hackerethik – Der Zugriff auf Wissen soll frei, dezentral, antibürokratisch und antiautoritär sein. Genau in diesen Forderungen hat die Hackerethik ihre anarchistischen Momente und ihre Bedingung ist der amerikanische Anarchismus. Während der Anarchismus in Europa längst verschwunden ist, hat er in der amerikanischen Tradition überdauert. Aus dieser Tradition heraus reagierten die ersten Hacker am MIT mit praktischem Anarchismus auf die Autoritäten, die ihren Zugang zum Computer beschränken wollten. Für Stallman war das Hackerparadies am MIT der lebendige Beweis dafür, dass eine anarchistische Gemeinschaft möglich ist. Nach ihrem Vorbild gründete er eine neue Hackergemeinschaft: Das GNU–Projekt. Die freie Software-Bewegung in Form von GNU, BSD oder *Open Source Initiative* ist die radikale, anarchistische Kritik an der heutigen Ordnung des „geistigen Eigentums". Im Gegensatz zu den BSD-Vertretern oder dem marktwirtschaftlichen Anarchismus Eric Raymonds plädiert Stallman für einen genossenschaftlichen Anarchismus, dass wir uns freiwillig zusammensetzen und ausdenken sollen, wie wir durch Zusammenarbeit für alle sorgen können.

1. Einleitung

In einem New Yorker Restaurant saßen zwei Männer beim Mittagessen zusammen und überlegten sich die nächsten Schritte ihrer kleinen Revolution. Einer der beiden, Eben Moglen, dachte kurz darüber nach, wie sie wohl auf die Leute wirken mochten,

die vorbei gingen: „Hier sind wir, zwei kleine bärtige Anarchisten, die sich ihre nächsten Schritte überlegen. Jeder, der unser Gespräch belauscht, wird denken, dass wir verrückt sind. Aber ich weiß: Ich weiß, dass wir hier an diesem Tisch eine Revolution vorbereiten." Und sein Gegenüber, Richard Stallman, sollte diese Revolution möglich machen (Vgl. Williams 2002, S. 184).

Doch nicht nur Eben Moglen, Rechtsprofessor an der Columbia Law School, sieht sich als Anarchist. Richard Stallman, eine der herausragendsten Figuren der freien Software-Bewegung, sieht sich selbst auch so. Wir dürfen uns die Anarchisten der freien Software dabei nicht klischeehaft als Chaoten mit zerzausten Haaren, irrem Blick und Armen voller Bomben vorstellen. Ganz im Gegenteil: Statt Chaos fordert Stallman eine neue Ordnung des „geistigen Eigentums" im Sinne der Hackerethik – der Zugriff auf Wissen soll frei, dezentral, antibürokratisch und antiautoritär sein. Stallman hat als Amerikaner weniger Scheu, sich als Anarchist zu bezeichnen, weil der Anarchie-Begriff in Amerika ein anderer ist als in Europa. Für David DeLeon, Professor für Geschichte an der Howard Universität in Washington D. C., ist der Anarchismus in Amerika in seinem Buch „The American as Anarchist" die einzig radikale konstruktive Kritik an der liberalen Gesellschaft der Vereinigten Staaten. Darüber hinaus, so meine These, ist der amerikanische Anarchismus Bedingung für die Hackerethik (Imhorst 2004).

Stallmans Botschaft ist eine radikal-politische Botschaft, denn es geht um Privateigentum, einen Eckpfeiler der Gesellschaft, in der wir leben. „Geistiges Privateigentum" in Form von Software ist die Gelddruckmaschine des ausgehenden 20. und des beginnenden 21. Jahrhunderts. Schließlich hat der reichste Mann der Welt, Bill Gates, seinen Reichtum weder mit Öl, Gold oder Aktienspekulationen erworben, sondern mit Software. Mit Copyrights und Patenten auf „geistiges Eigentum" in Form von Software kann man seit den 1980er Jahren Milliarden US-Dollar verdienen.

Die Gegner von Stallman werfen ihm vor, dass er das „geistige Eigentum" abschaffen wolle und mit seiner freien Software-Bewegung einer kommunistischen Utopie nachhänge. Er selbst sieht sich nicht als Kommunist oder antikapitalistischer Staatsfeind, der das Eigentum abschaffen will. Stallmans Lizenz, die GNU General Public License (GPL), ein politischer Ausdruck des *free spirit* in der freien Software–Bewegung, spricht auch nicht von der Abschaffung des „geistigen Eigentums". Im Gegenteil, sie will bestimmtes „geistiges Eigentum" schützen.

Der Programmierer freier Software verschenkt mit der GPL die Kontrolle über sein Werk, nicht aber das Werk als solches. Er behält die Autorenschaft über sein Programm. Dem Benutzer der Software wiederum werden bestimmte Freiheiten gewährt, wie die Freiheit, das Werk zu modifizieren und verändert zu veröffentlichen. An diese Freiheit ist nur eine Bedingung geknüpft: Das veränderte Werk muss wieder unter der GPL stehen. Ähnliche Lizenzen gibt es auch für Bücher, Musik und andere Formen „geistigen Eigentums". Diese Freiheiten können von keinem zurückgenommen werden. Freie Software soll nicht Eigentum eines Einzelnen, sondern das Eigentum von allen sein. Ihr Gegenstück ist die proprietäre Software. Ein proprietäres Programm wie Microsoft Word ist Privateigentum der Firma Microsoft. Wer sich Word installiert, hat nur ein Nutzungsrecht an dem Programm. Die umfangreiche Lizenz soll Word

davor schützen, dass das Programm einfach weitergegeben oder modifiziert wird. Die GPL dagegen ermutigt den Anwender zur Modifikation und zur Weitergabe der Software. Niemand ist vom Eigentum an GPL-Software ausgeschlossen. Ihre Verbreitung kann deshalb von niemandem kontrolliert werden. Wer sie haben möchte, kann sie einfach kopieren und weitergeben, wodurch die Verfügbarkeit von GPL-Software sehr schnell wächst. Die GPL verhindert zwar, dass Menschen von dem Gebrauch freier Software ausgeschlossen werden, aber sie schließt auf der anderen Seite ebenfalls aus, dass jemand aus freier Software proprietäre macht. Niemand kann daran gehindert werden, das freie Betriebssystem GNU/Linux zu benutzen, und niemandem kann es weggenommen werden. Jeder, der GNU/Linux aus dem Internet herunter lädt, auf seinen Rechner installiert, Kopien davon verschenkt oder verkauft, dem gehört es auch. In diesem Sinne ist die GPL eher eine Anti-Lizenz, weshalb Stallman von ihr auch lieber als *Copyleft* spricht anstatt von einem *Copyright*.

Grundlegend für Stallmans politische Philosophie ist die Hackerethik. Ein Kodex, den sich eine Gruppe von Computerfreaks am Massachusetts Institute of Technology (MIT) gegen Ende der 1950er Jahre gegeben hatte. Sie lernten gemeinsam, die ersten Computer am MIT zu programmieren und das Wissen darüber miteinander zu teilen. Das gemeinsame Programmieren, Lernen und den freien Austausch von Wissen nannten sie hacken und sich selbst Hacker, bevor Journalisten Computerpiraten so nannten. Die Hackerethik hat ihre anarchistischen Momente in den Forderungen nach Freiheit und Dezentralisierung, sowie in ihren antibürokratischen und antiautoritären Bestrebungen.

Während der Anarchismus in Europa weitestgehend verschwunden ist, hat er in der amerikanischen Tradition überdauert. Dafür macht DeLeon in „The American as Anarchist" historisch drei wesentliche Eigenschaften der amerikanischen Lebensweise verantwortlich: Den radikalen Protestantismus als nach innen gekehrte Religion, das weite Siedlungsgebiet, in denen sich Gemeinschaften der Kontrolle des Staates entziehen konnten und den amerikanischen Anarchokapitalismus:[1]

> „Unsere Radikalen konzentrieren sich auf Emanzipation, sie wollen die Ketten der Herrschaft zerreißen, anstatt neue zu schmieden. Sie sind Befreier und keine Gründer von Institutionen; sie treten für die Rechte von Frauen, Schwulen, Schwarzen und für die Befreiungstheologie ein; sie sind Propheten, keine Priester; Anarchisten und keine Verwalter. Im allgemeinen vermuten sie, dass der befreite Geist wenig bis gar keine Führung braucht." (DeLeon 1978, S. 4)

2. Amerikanischer Radikalismus

Der Anarchismus in den USA ist nach zwei Jahrhunderten Unabhängigkeit fundamental vom europäischen oder russischen Anarchismus zu unterscheiden. Die Einwohner der USA gaben sich eine eigene nationale Identität, als sie sich von Europa emanzipierten. Dabei schufen sie einen eigenen liberalen Radikalismus des „new lands, new

1 Alle Übersetzungen aus englischen Quellen durch den Autor.

men, new thoughts“. Der amerikanische Radikalismus war neu und keine Variation des europäischen Radikalismus. Die amerikanischen Anarchisten wollten niemals alle Autorität abschaffen. Sie waren Vertreter einer neuen Form von Ordnung, die des amerikanischen Anarchismus. Problematisch an dem Begriff des Anarchismus ist, dass er selbst niemals eine Doktrin oder feststehende Lehre sein kann. Der Anarchismus kann von jedem seiner Anhänger neu überdacht und wieder anders vertreten werden. In Amerika führte das zu der ausgeprägten Spaltung in einen „rechten“ und einen „linken“ Anarchismus. Konsistenz in der politischen Theorie darf man von Anarchisten nicht erwarten, denn der Anarchismus ist eher eine sich stets neu vollziehende Situationsanpassung. Doch gerade diese Anpassungsfähigkeit führt dazu, dass uns der Anarchismus durch die Geschichte der Menschheit begleitet und zuletzt verstärkt in der Hippie-Bewegung zum Ausdruck kam.

Die anarchistischen Hippies in Kalifornien waren die Pioniere der politischen Gegenkultur der 1960er Jahre. Sie beeinflussten die linken Bewegungen auf der ganzen Welt. Mit ihrer politischen Form der „Direkten Aktion“ – der ältere Begriff aus der anarchistischen Tradition ist „Propaganda der Tat“ – organisierten sie Kampagnen gegen Militarismus, Rassismus, sexuelle Diskriminierung und so fort. Die englischen Soziologen Richard Barbrook und Andy Cameron bezeichnen sie in ihrem Aufsatz „Kalifornische Ideologie“ als „Liberale im sozialen Sinne des Begriffs.“ Die Hippie–Bewegung schuf keine Hierarchien wie die traditionelle Linke, sie schufen kollektive und demokratische Strukturen.

> „Überdies verband die kalifornische Linke den politischen Kampf mit einer Kulturrebellion. Anders als ihre Eltern weigerten sich die Hippies, nach den strengen gesellschaftlichen Konventionen zu leben, in die organisierte Menschen seitens des Militärs, der Universitäten, der Unternehmen und selbst der linksgerichteten politischen Parteien gezwungen wurden. Statt dessen zeigten sie ihre Ablehnung der ordentlichen Welt durch lässige Kleidung, ihre sexuelle Promiskuität, ihre laute Musik und ihre entspannenden Drogen.“ (Barbrook und Cameron 1997)

3. Anarchistische Hacker

Für sexuelle Promiskuität, laute Musik und entspannende Drogen waren die ersten Computerfreaks am MIT nicht unbedingt zu gewinnen. Trotzdem haben sie einiges mit den Hippies gemeinsam, nämlich die anarchistische Ablehnung von autoritären und bürokratischen Strukturen und die Forderung nach ihrer Überwindung. Nach Steven Levys Buch „Hackers – Heroes of the Computer Revolution“ entwickelte sich die Subkultur der Hacker in den späten 1950er Jahren am MIT. Im Frühling 1959 bot die Universität den ersten Kurs in Computerprogrammierung an. Einen Computer zu bedienen war zu der Zeit mit großem Aufwand verbunden. Befehle gab man in diese riesigen Maschinen noch über Lochkarten ein, Bildschirme hatten sie nicht. Bis man allerdings so weit war, dass man einen Computer mit Lochkarten programmieren durfte, musste man zuerst an den Ingenieuren vorbei, die sich selbst als Priesterschaft

bezeichneten und über den Computer wachten. Wollten die ersten Hacker wie Peter Samson, Bob Saunders oder Alan Kotok an einen dieser Millionen Dollar teuren IBM-Computer, wurden sie sehr schnell von der Priesterschaft vertrieben.

> „An den IBM-Computern arbeiten zu wollen, war frustrierend. Es gab nichts schlimmeres, als die lange Wartezeit zwischen der Eingabe der Lochkarten und der Ausgabe des Ergebnisses. Wenn du nur einen Buchstaben in einer Instruktion falsch gesetzt hast, ist das ganze Programm abgestürzt und du musstest den ganzen Prozess von vorne beginnen. Das ging Hand in Hand mit der erdrückenden Ausbreitung gottverdammter Regeln, die die Atmosphäre des Rechenzentrum immer weiter durchdrangen. Die meisten Regeln wurden extra erfunden, um junge verrückte Computerfans wie Samson und Kotok und Saunders physisch von den Rechnern fernzuhalten. Die strengste Regel war, dass niemand den Computer wirklich berühren oder sich an ihm zu schaffen machen durfte. Das war etwas, was sie wahnsinnig machte." (Levy 1984, S. 27)

Samson und Kotok reichte es nicht, sich die Computer nur anzusehen. Sie wollten wissen, wie sie funktionierten. So belegten sie den Computerkurs, um endlich herauszufinden, wie Computer arbeiten. Die strengen Regeln, die im Bereich der Lochkartencomputer herrschten, und die Priesterschaft, die über die Rechner wachte, machten das Hacken schwer. In Gang kam die Subkultur der Hacker erst mit einer neuen Computergeneration, die keine Lochkarten mehr brauchte. Wollte man an den neuen Computern arbeiten, gab es auch nicht so eine große bürokratische Hürde zu überwinden wie bei den alten IBM-Maschinen. Dadurch, dass man Programme direkt mit Tastatur und Bildschirm am Computer starten konnte, ohne einen Stapel Karten einlesen zu müssen, inspirierten die neuen Computer die Programmierer zu einer neuen Form des Programmierens, und die Hacker waren ihre Pioniere. An einem Computer wurden außerdem mehrere Bildschirme und Tastaturen angeschlossen, so dass mehrere Leute den Rechner nutzen konnten, indem sie sich die Rechenzeit teilten.

4. Hackerethik

In ihrem täglichen Kampf um Rechenzeit und gegen die Autoritäten, die sie daran hinderten, zu programmieren, entwickelten die jungen Hacker ihre eigene Ethik. Noch waren sie wenige, und sie nahmen ihre Hackerethik noch nicht ganz so ernst, wie es später bei der Fall sein wird. Die Hackerethik wurde nicht als Manifest veröffentlicht, sie wurde in den ersten Jahren mündlich überliefert. Sie wurde auch nicht diskutiert, sondern von den Hackern, die sie annahmen, wie Axiome hingenommen. Die wichtigsten Punkte der Hackerethik sind, dass der Zugriff auf Computer unbegrenzt, total und alle Information frei sein soll. Autoritäten soll misstraut und Dezentralisierung gefördert werden (Levy 1984, S. 40 ff.).

Vor allem die Universitätsbürokratie hat es Hackern sehr schwer gemacht, wertvolle Rechenzeit an den wenigen Computern zu bekommen. Offene Systeme ohne Büro-

kratie und Autoritäten ermöglichten es ihnen, viel produktiver an den Computern zu sein. Ohne autoritäre Aufsicht durch die IBM-Priester konnten sie am Computer viel mehr leisten. Sobald sie hinter der Konsole einer IBM-Maschine saßen, hatten sie Macht über sie. So ist es fast natürlich, dass sie jeder anderen Macht misstrauten, die sie ohnmächtig machen und die Hacker davon abhalten wollte, ihre Macht über den Computer voll auszunutzen.

Levy hält in „Hackers" die Geschichte der Hackerkultur und ihrer Hackerethik am MIT bis zu ihrem damals vorläufigen Ende 1984 fest. Ein ganzes Kapitel befasst sich mit Richard Stallman, den Levy als den letzten wahren Hacker sieht. Dort sagt Stallman, dass die Hackerkultur am MIT das lebende Beispiel für eine anarchistische und großartige Einrichtung gewesen sei, bevor sie ausgelöscht wurde (Levy 1984, S. 423). Stallman gründete nach dem Vorbild der Hackerkultur eine neue Gemeinschaft, das GNU-Projekt, einer der wichtigsten Pfeiler der freien Software-Bewegung.

Denn die freie Software-Bewegung ist mehr als das GNU-Projekt. Die wichtigste Trennlinie verläuft wohl zwischen den Anhängern von BSD- und GPL-Lizenzen. Die Berkeley Software Distribution (BSD) ist eine Version des Betriebssystems Unix, die an der Universität von Kalifornien in Berkeley Ende der 1970er Jahre entstanden ist. BSD bezeichnet heute eine ganze Reihe von Unix-Ablegern, wie FreeBSD, NetBSD oder OpenBSD. Im Gegensatz zur GPL erlaubt es die BSD-Lizenz, dass der freie Quellcode unter bestimmten Bedingungen in proprietärer Software verwendet werden darf. Es musste bis vor kurzem lediglich der Universität von Kalifornien gedankt werden. Am Anfang eines Software-Projekts kann es eine große Streitfrage sein, ob es die GPL- oder die BSD-Lizenz haben soll. Entscheidet man sich für die BSD-Lizenz, gibt man anderen die Möglichkeit, kommerzielle Versionen aus der eigenen Arbeit zu entwickeln. Entscheidet man sich für die GPL, ist jeder verpflichtet, Quellcode in das Projekt zurückfließen zu lassen. Der Journalist Peter Wayner schreibt in seinem Buch „Kostenlos und Überlegen! Wie Linux und andere freie Software Microsoft das Fürchten lehren" über diesen Streit:

> „Wer sich der GPL anschließt, hat wahrscheinlich auch weniger Probleme mit Richard Stallman, oder sieht zumindest davon ab, öffentlich über ihn herzuziehen. GPL-Anhänger neigen zur individualistischen Bilderstürmerei, halten die eigenen Projekte für ziemlich kultig und werden von einer merkwürdigen Mischung aus persönlicher Überzeugung und 'Was-bin-ich-doch-für-ein-cooler-Typ'-Hysterie angetrieben. Anhänger von BSD-Lizenzen machen dagegen einen eher pragmatischen, organisierten und konzentrierten Eindruck." (Wayner 2001, S. 159)

Die BSD-Anhänger treiben kaum einen Kult um ihre Lizenz. Meist streichen sie nur die Freiheit ihrer BSD-Lizenz gegenüber der GPL-Lizenz heraus. Sie haben auch keine Coverstars wie Richard Stallman oder Linus Torvalds. Von der Presse werden die BSD-Projekte deshalb auch meist ignoriert. Stallmans Kreuzzug zur Befreiung des Quellcodes können BSD-Anhänger daher nicht viel abgewinnen.

Stallmans Sicht ist radikaler. Er will ein System freier Software, das Unix zwar ähnlich, aber besser sein soll. Deshalb nannte er seine Arbeit GNU, ein rekursives

Akronym für „'GNU's Not Unix". Ziel des GNU-Projekts ist es seitdem, ein vollständig freies und funktionstüchtiges Betriebssystem mit allen notwendigen Programmen zu entwickeln. Es sollte von Anfang an mehr als nur ein Sammelbecken für freie Software sein. GNU ist ein System freier Software, das jede proprietäre Software durch GNU-Software ersetzen soll. Mit der Gründung des GNU-Projekts beginnt Stallmans Kreuzzug. Die Freiheit, die vorher nur in der Hackerethik kodifiziert war, wurde nun in einem Vertrag zwischen dem Autor und dem Nutzer rechtsverbindlich in der GNU General Public License festgehalten. Die GPL soll das System freier Software davor schützen, ausgebeutet zu werden.

5. Der Anarchismus der *Open Source Initiative*

Am 15. Mai 1969 stürmten bewaffnete Polizeieinheiten auf Befehl des Gouverneurs von Kalifornien, Ronald Reagan, den People's Park in der Nähe des Campus der Universität von Kalifornien, um gegen protestierende Hippies vorzugehen. Dabei wurde ein Mensch getötet und über hundert verletzt. Das konservative Establishment mit Gouverneur Reagan und die Gegenkultur der Hippies schienen zwei antagonistische Gegensätze zu sein. David DeLeon findet in seinem Buch „The American as Anarchist" allerdings heraus, dass Gouverneur Reagan und die Hippies eher zwei Extreme desselben amerikanischen Anarchismus sind.[2] Für DeLeon ist der Anarchismus in den USA die einzige radikale Kritik von Rechten und Linken an der liberalen amerikanischen Gesellschaft. Er nennt die beiden Flügel auch „rechte und linke Libertäre (*libertarians*)". Wobei Libertär nur ein anderes Wort für Anarchist ist.

Wendet man die Theorie von DeLeon auf die Anhänger der GPL und die Verfechter der Open-Source-Definition an, dann sind sie ebenfalls zwei Extreme des amerikanischen Anarchismus. Eric S. Raymond nennt diese beiden Extreme in seinem Aufsatz „The Cathedral and the Bazaar" das Kathedralen- und das Basar-Modell (Vgl. Raymond 2001).

Frederick P. Brooks stellte in seinem 1975 erschienen Buch „The Mythical Man–Month: Essays on software engineering" (Vgl. Brooks 1995) ein Gesetz auf, nachdem sich jedes Softwareprojekt verzögert, je mehr Entwickler an dem Projekt beteiligt sind. Wie viele andere Hacker glaubte auch Eric Raymond als ehemaliges Mitglied des GNU-Projekts, dass viele Köche den Brei verderben würden und nach dem Brook'schen Gesetz ein Software-Projekt davon profitiere, wenn nur wenige an ihm beteiligt seien. Auch die Software-Projekte des GNU-Projekts bestehen nach diesem Gesetz aus nur wenigen Entwicklern. Zu Raymonds Erstaunen bewies Linus Torvalds mit der schnellen Veröffentlichung von Linux das genaue Gegenteil: Je mehr Hacker er einlud, im Linux-Projekt mitzumachen, desto besser wurde es.

Raymond schrieb seine Beobachtungen in dem Aufsatz „The Cathedral and the Bazaar" nieder, indem er die unterschiedlichen Stile in der Leitung von GNU-Projekten mit dem Linux-Projekt kontrastierte. Den Aufsatz entwickelte er aus einer Rede, die

2 In seinem Wahlkampf zum Präsidenten trat Reagan mit anarchokapitalistischen Wahlkampf-Slogans entschieden für das freie Unternehmertum und für eine weitere Deregulierung des freien Marktes ein.

er zum ersten Mal 1997 auf einem Linux-Kongress in Deutschland hielt. Der Titel des Aufsatzes stammt von seiner zentralen Analogie: GNU-Programme seien eher wie beeindruckende Kathedralen, zentral geplante Monumente der Hackerethik, gebaut für die Ewigkeit. Das Linux-Projekt gleiche eher einem großen Basar mit schwatzenden Händlern. In dieser Analogie impliziert ist auch ein Vergleich zwischen Stallman und Torvalds. Stallman sei der klassische Vertreter des Architekt einer Kathedrale. Er ist ein Programmier-Guru, der für 18 Monate verschwinden könne, um mit einem genialen C-Compiler wieder aufzutauchen. Torvalds sei mehr wie der Gastgeber einer Dinnerparty. Diskussionen über das Design von Linux überlässt er den einzelnen Projektgruppen. Er schreitet nur ein, wenn es in einer Gruppe dermaßen Streit gibt, dass sie einen Schiedsrichter braucht. Schließlich entscheidet letztendlich Torvalds darüber, was in den Kernel hinein kommt. Seine wichtigste Aufgabe bestehe darin, den Fluss der Ideen aufrechtzuerhalten.

Seine Analyse brachte Raymond den Ruf ein, ein „Evangelist des freien Marktes" zu sein. Er hält von staatlichen Eingriffen in den Markt nicht viel. Der Einzelne soll generell durch Dereglementierung gestärkt werden, was bei ihm auch Waffenbesitz miteinschließt. DeLeon würde ihn in „The American as Anarchist" eben seine Kategorie der „rechten Libertäre" einsortieren. „Rechte Libertäre" sind Anarchisten, die meinen, dass die Regierung sie in Ruhe lassen solle, so dass sie mit ihrem Geld und ihren Waffen anfangen können, was sie wollen. Als Raymond nun die freie Software-Bewegung und ihren produktiven Anarchismus untersuchte, entdeckte er, was er als rechter Libertarier entdecken wollte: einen unreglementierten freien Markt. Die Grundlage des Erfolges der freien Software-Bewegung ist demnach die Freiheit der Nutzer. Das Basar-Modell spricht somit schon für eine größtmögliche Deregulierung und Freiheit, da viele verschiedene Händler miteinander konkurrieren. GNU-Projekte, oder auch firmeneigene Entwicklungen proprietärer Software, sollen dagegen ähnlich strukturiert sein wie die mittelalterliche Gemeinde: Der Kathedralenbau wird mit dem Geld der Stadt von einer Gruppe von Priestern vorangetrieben, um die Ideen eines Architekten zu verwirklichen. So kann der Kathedralenbau nur gelingen, wenn genügend Geld, ein talentierter Architekt und Arbeiter vorhanden sind. Die vielen verschiedenen Händler auf dem Basar versuchen dagegen, sich gegenseitig aus dem Feld zu schlagen. Der Beste von ihnen hat auch die meisten Kunden, ganz im sozialdarwinistischen Sinn: Der am besten Angepasste überlebt.

Das Problem an Raymonds Aufsatz ist allerdings, dass weder GNU-Projekte als reine Kathedralen noch das Linux-Projekt wirklich als Basar erscheinen. Vor allem dem Linux-Projekt mit seinen Veröffentlichungen in möglichst kurzen Zeiträumen und seinen Tausenden von Mitarbeitern, steht ganz oben voran Linus Torvalds, der entscheidet, was in den neuen Kernel Einzug findet und was nicht. Das Linux-Projekt ist mehr eine Mischform als ein reiner Basar oder eine reine Kathedrale.

In seiner Rede sprach Raymond 1997 noch von freier Software. Ab Februar 1998 ersetzte er in seinem Aufsatz „freie Software" durch „Open Source". Für ihn und einige andere Anhänger der freien Software-Bewegung wurde Stallman immer mehr zum Ärgernis. Sie fanden, dass Stallman für Geschäftsleute wegen seiner politischen Aussagen zu sehr nach Kommunismus roch. Außerdem wollten sie, dass die Bewegung

sich nicht zu sehr auf die GPL konzentrierte. Sie wollten ein System von Software, in das GPL-Software genauso passt wie Software, die unter der BSD- oder ähnlichen Lizenzen fällt und nannten das Ganze Open Source. Volker Grassmuck sagt in „Freie Software – Zwischen Privat- und Gemeineigentum" dazu:

> „'Free' ist nicht nur zweideutig ('Freibier' und 'Freie Rede'), sondern offensichtlich war es in *The Land of the Free* zu einem unanständigen, 'konfrontationellen', irgendwie kommunistisch klingenden *four-letter word* geworden." (Grassmuck 2002, S. 230)

6. Der letzte wahre Hacker

Levy interviewte für sein Buch „Hackers" auch Richard Stallman. Seiner Geschichte widmet er ein ganzes Kapitel, was nicht umsonst mit „Epilogue: The Last Of The True Hackers" überschrieben ist. Denn 1984 stand es schlecht um die freie Software. Stallman gehört zur Generation der ersten Hacker, die riesige IBM-Maschinen an amerikanischen Universitäten programmiert hatten. Die jungen Leute, die in den 1980er Jahren in den Computerräumen der Universitäten auftauchten, lernten an ihren Heimcomputern unberührt von jeder Hackerethik und Hacker-Gemeinschaft das Programmieren.

> „Diese neuen Leute schrieben wie ihre Vorgänger im Rechenzentrum aufregende Programme. Doch mit ihnen hielt auch etwas neues Einzug – als ihre Programme auf den Computermonitoren auftauchten, waren sie mit Copyrights versehen. Für Stallman, der daran festhielt, dass alle Information frei fließen sollte, war das Blasphemie. 'Ich denke nicht, das Software jemanden gehören sollte', da diese Praxis die Menschlichkeit als Ganzes sabotiert. Es hindert Leute daran, den kompletten Nutzen aus der Software zu ziehen." (Levy 1984, S. 419)

Diese neuen Hacker interessierten sich auch nicht sonderlich für die Hackerethik. Stallman hatte im Rechenzentrum des MIT gelernt, dass eine anarchistische Institution möglich ist. Nur fehlten ihm die Mitstreiter aufgrund der Dezentralisierung der Hacker durch die Heimcomputer. Anfang der 1980er Jahre fühlte er sich als letzter Anhänger einer scheinbar toten Bewegung, die sich an den anarchistischen Grundsätzen der Hackerethik ausrichtete. Diese Bewegung wollte er wiederbeleben. Mit der freien Software-Bewegung wird die Hackerkultur wiedergeboren und Stallman tritt an, den Quellcode von proprietären Lizenzen zu befreien.

Die freie Software-Bewegung in Form von GNU, BSD oder *Open Source Initiative* ist die radikale, anarchistische Kritik an der heutigen Ordnung des „geistigen Eigentums" nicht nur in der liberalen Gesellschaft der Vereinigten Staaten, sondern an dessen Ordnung in der ganzen globalisierten Welt. Im Gegensatz zu den BSD-Vertretern oder dem marktwirtschaftlichen Anarchismus Eric Raymonds von der *Open Source Initiative* plädiert Stallman für einen genossenschaftlichen Anarchismus, der

zumindest in Bezug auf das „geistige Eigentum“ frei nach dem französischen Anarchisten Jean-Pierre Proudhon sagt, dass Eigentum Diebstahl sei. Für viele ist die Forderung nach Abschaffung des geistigen Privateigentums heute undenkbar. Dabei war noch vor einem halben Jahrtausend die Einführung von Eigentum undenkbar. So sagt Jeremy Rifkin in „Access – Das Verschwinden des Eigentums“:

> „Dass wir Marktsystem und Warentausch hinter uns lassen, [. . .] ist derzeit für viele Menschen noch genauso unvorstellbar, wie es die Einhegung und Privatisierung von Land und Arbeit und damit ihre Einbindung in Verhältnisse des Privateigentums vor einem halben Jahrtausend gewesen sein mögen.“ (Rifkin 2000, S. 24)

Stallman und die GNU-Fraktion der freien Software-Bewegung wollen „geistiges Eigentum“ nicht nur in Gestalt von Software, sondern auch in der Gestalt von Büchern oder Musik von proprietären Lizenzen befreien. Warum, das sagt Stallman im Gespräch mit Spiegel Online: „Ich tendiere mehr zu der linken anarchistischen Idee, dass wir uns freiwillig zusammensetzen und ausdenken sollen, wie wir durch Zusammenarbeit für alle sorgen können“ (Klagges 1996).

Literaturverzeichnis

Barbrook, R. und Cameron, A. (1997), ‘Die kalifornische Idiologie’, Telepolis, http://www.telepolis.de/deutsch/inhalte/te/1007/1.html/ [30. Nov 2004].

Brooks, F. P. (1995), *The Mystical Man-Month: Essays on software engineering*, Addison-Wesley, New York.

DeLeon, D. (1978), *The American as Anarchist*, John Hopkins University Press, Baltimore, MA.

Grassmuck, V. (2002), *Freie Software – Zwischen Privat- und Gemeineigentum*, Bundeszentrale für politische Bildung, Bonn.

Imhorst, C. (2004), *Die Anarchie der Hacker – Richard Stallman und die Freie-Software-Bewegung*, Tectum Wissenschaftsverlag, Marburg.

Klagges, H. (1996), ‘Es reicht mir nicht, nur einfach neugierig auf die Zukunft zu sein, ich will etwas ändern. – Interview mit Richard Stallman’, Klagges.com, http://www.klagges.com/pdf/interview_stallman.pdf [30. Nov 2004].

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

Raymond, E. S. (2001), *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*, 1. Aufl., O’Reilly, Sebastopol, CA.

Rifkin, J. (2000), *Access – Das Verschwinden des Eigentums*, Campus Sachbuch, Frankfurt am Main.

Wayner, P. (2001), *Kostenlos und Überlegen! Wie Linux und andere freie Software Microsoft das Fürchten lehren*, DVA, Stuttgart/München.

Williams, S. (2002), *Free as in Freedom – Richard Stallman’s Crusade for Free Software*, O’Reilly, Sebastopol, CA.

Freie Software und Freie Gesellschaft

Stefan Merten und Stefan Meretz

Der weltweite Erfolg freier Software hat die Frage aufgeworfen, ob ihre Prinzipien verallgemeinerbar und auf die gesamte Gesellschaft übertragbar sind. 1999 gründete sich das Oekonux-Projekt rund um diese Frage. Oekonux steht für „Oekonomie und GNU/Linux". Zahlreiche Analysen aus unterschiedlichen Richtungen haben zu Thesen geführt, die viele für bahnbrechend halten und an denen sich viele reiben – innerhalb und außerhalb des Projektes. Der erste Teil des Beitrages beschäftigt sich mit den Eigenschaften freier Software und dessen Prinzipien. So wird zuerst die Produktionsweise von freier Software betrachtet, der Unterschied zwischen einfach und doppelt freier Software gezeigt und anschließend erklärt, warum es sich bei freier Software um eine neue Qualität der Produktivkraftentwicklung handelt. Diese Betrachtungen sollen im letzten Teil dazu dienen, in freier Software eine Keimform für eine neue Gesellschaft zu sehen. Diese Betrachtung wird mit dem Bild der „GPL-Gesellschaft" abgeschlossen.

1. Einleitung

Freie Software hinsichtlich eines möglichen Ausgangspunkts einer neuen, fundamental veränderten Gesellschaftsformation zu untersuchen, hat sich das Projekt Oekonux zur Aufgabe gemacht. Das ganz überwiegend virtuelle Projekt gruppiert sich um mehrere Mailing-Listen und einige Websites. Im nicht-virtuellen Raum trat das Projekt mit bisher drei internationalen Konferenzen an die Öffentlichkeit.[1] Der vorliegende Artikel, der diese zentralen Thesen beleuchtet, wurde in einem offenen Prozess entwickelt,[2] der auch jetzt noch für Beiträge offen ist. Der Artikel steht nicht nur zum Lesen zur Verfügung, sondern kann auch direkt online im Browser Absatz für Absatz kommentiert werden.

1 Informationen zum Projekt und den bisherigen Konferenzen befinden sich unter http://www.oekonux.de/projekt/index.html.

2 Der Beitrag befindet sich online unter unter http://www.opentheory.org/ox_osjahrbuch_2005/.

2. Produktionsweise freier Software

Die Produktionsweise freier Software unterscheidet sich grundsätzlich von der proprietärer Software. Dies betrifft weniger die technischen Verfahren, sondern vor allem die individuelle Motivation und die soziale Organisation. Diese Produktionsweise ist gekennzeichnet durch *Wertfreiheit*, *Selbstorganisation*, *Globalität* und *Selbstentfaltung*. Diese vier zentralen Begriffe sollen einleitend kurz umrissen und im restlichen Text aus verschiedenen Perspektiven weiter erläutert werden (Vgl. Merten 2000, Meretz 2000*b*, 2004).

Die Entwicklung von Software ist mit Aufwand verbunden. Bei freier Software wird dieser Aufwand in der Regel jedoch nicht (monetär) entlohnt. Wie auf vielen anderen Gebieten menschlichen Lebens strengen sich die Menschen hier aus anderen Gründen an, als Geld dafür zu erhalten. Das Resultat dieser Tätigkeit ist deswegen ökonomisch *wertfrei* und unterscheidet sich damit wesentlich von der wertbasierten Arbeit, die auf die Erreichung von Lohn oder Profit abzielt (Vgl. Meretz 2000*a*).

Die sozialen Organisationsformen freier Software sind so verschieden wie die Projekte selbst. Niemand gibt von außen vor, wie etwas zu sein hat. Jedes Projekt organisiert sich selbst und findet die Form, die ihm gemäß ist, oft einfach durch Ausprobieren.

Die *globale Vernetzung* istdas Resultat der Möglichkeiten des Internets. Jedes noch so kleine Projekt, das sich einer der vielen frei zugänglichen Projekt-Infrastrukturen bedient (SourceForge, Savannah etc.) oder selbst eine betreibt, ist weltweit verfügbar. Menschen, die sich noch nie gesehen haben und vielleicht auch niemals sehen werden, können so zusammen etwas Nützliches erschaffen. Ohne das Internet mit seinen vielfältigen Diensten wie E-Mail, FTP oder WWW wäre freie Software in ihrer heutigen, entfalteten Form nicht denkbar.

3. Hauptantrieb Selbstentfaltung

Selbstentfaltung ist ein zentraler Begriff für das Verständnis freier Software. Selbstentfaltung meint nicht einfach „Spaß haben“ und besitzt eine individuelle und eine gesellschaftliche Dimension. Individuell meint Selbstentfaltung das persönliche Entfalten der eigenen Möglichkeiten, mithin das Entwickeln der eigenen Persönlichkeit. Eine so verstandene Entfaltung der Persönlichkeit hat verschiedene Formen der Entäußerung: produktive, reproduktive, technische, kulturelle, kommunikative, konsumtive etc. Diese können für andere nützlich sein (Vgl. Himanen 2001).

Die gesellschaftliche Dimension der Selbstentfaltung betrifft die Abhängigkeit der eigenen Entfaltung von der Entfaltung der anderen. Ich kann mich nur entfalten, wenn die anderen es auch tun. Die anderen – potenziell alle anderen – sind meine Entfaltungsbedingung, wie ich umgekehrt Entfaltungsbedingung für die anderen bin. Es entsteht eine positive Rückkopplung: Mein Bestreben richtet sich darauf, dass die anderen sich entfalten können, damit ich mich entfalten kann. Würde ich mich nur darauf konzentrieren, was ich zu tun wünsche und die anderen ignorieren oder gar ausgrenzen, dann würde ich mir selbst schaden.

Diese Dynamik können wir, mehr oder weniger ausgeprägt, bei freier Software beobachten. Die positive Rückkopplung kommt zustande, weil und wenn es keine dritten, entfremdeten Gründe gibt, tätig zu werden. Da freie Software nicht für den Verkauf produziert wird, gibt es keine entfremdeten Gründe, sondern nur jeweils meine Gründe, freie Software zu entwickeln oder zu unterstützen. Proprietäre Software hingegen wird für einen dritten, fremden, der Software äußerlichen Zweck entwickelt.[3]

Im Projekt Oekonux wurde diese Analyse zu dem Satz verdichtet: „Die Selbstentfaltung des Einzelnen ist die Bedingung für die Entfaltung Aller – und umgekehrt". Besonders deutlich wird die Unterscheidung, wenn man den gleichen Satz für die entfremdete Warenproduktion formuliert. Dort gilt: Die Entwicklung des Einzelnen ist möglich auf Kosten der Entwicklung der anderen – und umgekehrt.

Selbstentfaltung darf nicht mit Selbstverwirklichung verwechselt werden. Während Selbstentfaltung die anderen als Bedingung für die eigene Verwirklichung versteht, blendet Selbstverwirklichung die gesellschaftliche Dimension aus. Selbstverwirklichung ist statisch und begrenzt, sie geht von einer Anlage aus, die verwirklicht werden will und endet mit der Verwirklichung. Selbstentfaltung hingegen ist dynamisch. Jede erreichte Entfaltung ist wiederum nur Bedingung und Möglichkeit neuer Formen der Entfaltung. Eine Gesellschaft der Selbstentfaltung wäre eine reiche Gesellschaft (Gorz 2004).

4. Einfach und doppelt freie Software

In der freien Software-Bewegung wird von verschiedenen Seiten immer wieder betont, dass freie Software Bestandteil von Geschäftsmodellen sein kann. Es soll möglich sein, mit freier Software Geld zu verdienen.

Nun ist klar, dass Geschäftsmodelle, die mit einer Verknappung eines fertigen Produkts operieren, bei freier Software nicht funktionieren können. Verknappung von Informationsgütern ist unter den Bedingungen der weltweiten digitalen Kopie allgemein nur zu erreichen, wenn den NutzerInnen das Recht genommen wird, selbst das Informationsgut weiterzugeben. Die Grundrechte freier Software – uneingeschränkte Einsatzmöglichkeiten, Einsicht in und Möglichkeit zur Anpassung der Quellen, unbeschränkte Weitergabe originaler oder veränderter Versionen – erlauben diese Verknappung jedoch nicht.

Es ist zwar nicht möglich, fertige Produkte direkt zu verkaufen, dennoch gibt es verschiedene Dienstleistungen rund um freie Software, die verkauft werden können: beispielsweise Wartung, Installation oder Zusammenstellen von Distributionen. Diese Geschäftsmodelle arbeiten zwar mit vorhandener, fertiger freier Software, verkauft wird aber letztlich die Dienstleistung. Kaufe ich beispielsweise eine Distribution von SuSE/Novell, so kaufe ich ein Handbuch, CDs und DVDs sowie das Recht auf telefonischen Support für die Installation, das sogar durch einen speziellen, individuellen Code abgesichert ist. Die freie Software, die auf den mitgelieferten Medien

3 Der Markt entscheidet, ob die Software überlebt und wenn sich zu wenige KäuferInnen finden, verschwindet die Software. Das gibt es bei freier Software nicht. Solange sich eine EntwicklerIn für die Software interessiert, gibt es sie – auch, wenn es keine aktuellen NutzerInnen mehr gibt.

enthalten ist, kann ich mir jedoch ganz legal und ohne Bezahlung auch direkt von den SuSE-FTP-Servern downloaden. Tatsächlich bezahlt wird SuSE also für die materiellen Produkte, die sich auf die verteilte freie Software beziehen sowie für Support-Dienstleistung. Grundlage für dieses Geschäftsmodell ist die Dienstleistung der Zusammenstellung und Pflege der Distribution aus dem riesigen Pool vorhandener freier Software. Auch die Anbieter verschiedener Merchandising-Produkte rund um einzelne freie Software-Projekte oder auch Bücher zu freier Software verkaufen nicht freie Software, sondern eben Plüschpinguine.

Ist mit fertiger freier Software selbst kein Geschäftsmodell zu begründen, so gilt das nicht für freie Software, die noch nicht existiert. Es sind durchaus Geschäftsbeziehungen möglich, bei denen freie Software *im Auftrag* erstellt wird. Solche Auftragsarbeiten unterscheiden sich im Falle von Projekten ohne Anbindung zu einer Community von proprietärer Software lediglich durch die Lizenz, unter der das fertige Produkt später steht. Ebenfalls in diese Kategorie fällt die Weiterentwicklung freier Software, die in Firmen für eigene Zwecke durchgeführt wird.

Gegenüber der klassischen freien Software, wie sie ein Richard Stallman oder ein Linus Torvalds entwickelten, gibt es bei den genannten Geschäftsmodellen jedoch einen wichtigen Unterschied. Wie, wohin und wie schnell sich freie Software entwickelt, die ohne externen Auftrag entsteht, liegt allein in der Entscheidung des jeweiligen Projekts. Zu den Freiheiten, die die Lizenzen den NutzerInnen gewähren, tritt in diesen Fällen die Freiheit der EntwicklerInnen, die nicht an Weisungen von Auftraggebern gebunden sind. In Fällen, wo allein die Selbstentfaltung der EntwicklerInnen den Fortgang des Projekts bestimmt, sprechen wir von *doppelt freier Software*.

Demgegenüber sprechen wir von *einfach freier Software*, wenn die EntwicklerInnen in ihren Entscheidungen nicht frei, sondern an einen Auftraggeber gebunden sind. Die EntwicklerInnen entfremden sich in solchen Projekten von ihrem Produkt, da sie auf Grund der Abhängigkeit von der Bezahlung Entscheidungen des Auftraggebers berücksichtigen müssen, die aus ihrer Sicht für das Produkt schädlich sein können. Alle, die schon einmal Software im Auftrag einer Firma hergestellt haben, kennen unzählige Beispiele für von Marketing oder Vertrieb bestimmte Terminpläne, technisch überflüssige Hochglanzfeatures usw. Hier zeigt sich deutlich die Entfremdung vom Produkt, die mit der Erfüllung des dem Produktnutzen äußerlichen Zwecks der Verwertung entsteht.

Betrachten wir freie Software als neue Produktionsweise, so tritt gerade die freie Entscheidung der EntwicklerInnen in den Vordergrund. Nicht getrieben von Marktvorgaben, mithin frei von den Zwängen der Verwertung, können sich die EntwicklerInnen auf die bestmögliche Qualität der Software konzentrieren. Unter anderem ist es möglich, wie jüngst bei GIMP geschehen,[4] sich eine zweijährige Auszeit zu nehmen, in

4 Aus c't 13/2004, S. 80: „Vor-Meilenstein – Open-Source-Bildbearbeitung GIMP: Fit für die Profi-Liga?" Doch die eigentliche Dreijahresleistung der Entwickler fand hinter den Kulissen statt, wie GIMP-Maintainer Michael „Mitch" Natterer gegenüber c't erklärte: „Mit dieser Version wurde die Programmlogik strikt vom Frontend getrennt, kein Quellcode-Stein blieb auf dem anderen. In GIMP 1.2 befanden sich sämtliche Quellcode-Dateien in einem Verzeichnis; niemand wusste genau, was wozu gehört. Kleine Änderungen an einer Stelle konnten anderswo verheerende Auswirkungen haben." Für die Entwickler

der die historisch gewachsene Code-Basis durch ein neues, qualitativ hochwertigeres Fundament ersetzt wird. Anstatt neue Features zu implementieren, wird hier die langfristige Qualität in einer Weise gesichert, wie man sie sich von bekannten proprietären Software-Produkten wünschen würde.

Der qualitative Vorsprung doppelt freier Software ist struktureller Natur und kann auf Grund der in einfach freier Software angelegten Entfremdung von dieser nicht eingeholt werden. Dieser qualitative Vorsprung ist es aber letztlich, der dem Produktivkraftmodell doppelt freier Software den entscheidenden Vorteil vor dem einfach freier oder proprietärer Software gibt. Gäbe es diesen Vorsprung nicht, so hätte freie Software bei Software-NutzerInnen keine Chance gegen proprietäre Software gehabt. Der heute zu beobachtende Erfolg freier Software basiert im Fundament nämlich auf der doppelt freien Software, die teilweise schon vor vielen Jahren geschrieben wurde.

Dennoch treibt auch einfach freie Software insgesamt die Entwicklung in Richtung einer freien Verfügung über freie Produkte voran und in der Praxis mischen sich beide Formen oft. Dies geschieht insbesondere dann, wenn eine doppelt freie Community zu EntwicklerInnen einfach freier Software hinzutritt.

5. Freie Software ist keine Ware

Wollen wir ermessen, inwiefern sich freie Software vom vorherrschenden Wirtschaftsmodell der Marktwirtschaft unterscheidet, so ist es sinnvoll, freie Software mit einem der zentralen Elemente der Marktwirtschaft zu vergleichen: *Der Ware*. Unter Waren verstehen wir in diesem Kontext Güter, die primär zum Zwecke des Verkaufs auf einem Markt produziert werden und sich also von Gütern unterscheiden, die primär aus anderen Gründen produziert werden, z. B. weil sie nützlich sind. Für am Markterfolg orientierte Produkte genügt bei den ProduzentInnen ein *relativer Qualitätsanspruch*, da es lediglich darum geht, in den Augen der avisierten KäuferInnen besser zu sein als die Konkurrenz. Während sich die Qualität von marktorientierten Produkten in dieser Relativität erschöpft und in Monopolsituationen zu deutlich sichtbarer mangelnder Qualität führt, liegt bei einer Produktion, bei der die Nützlichkeit eines Produkts im Vordergrund steht, ein *absoluter Qualitätsanspruch* in der Logik der ganzen Produktionsweise.

Für die Wareneigenschaft ist der Preis der Ware im Übrigen nicht relevant und preislose Waren sind uns spätestens seit den Lockangeboten für Handy-Verträge vertraut.[5]

---

stand außer Frage, dass sie mit dieser Struktur an einem toten Punkt angelangt waren. „Um an wirklich neue Features wie CYMK, 16 Bit oder Ebenengruppen überhaupt denken zu können, musste das Ganze erst einmal auf ein ordentliches Fundament gesetzt werden", konkretisiert Natterer die Probleme des alten GIMP. Heute befinden sich die internen Funktionen von GIMP in 12 klar voneinander getrennten Modulen. Die Oberfläche wurde fast komplett neu geschrieben und ebenfalls in Module aufgeteilt. „Dabei hat fast jede Zeile Quellcode mehr als einmal ihren Platz gewechselt", umreißt Natterer das Mammutprojekt.

5 Ein Sonderfall ist Freeware, die keine freie Software ist. Freie Software zeichnet sich durch freie Verfügung, freie Quellen, freie Änderbarkeit und freie Verteilbarkeit aus. Freeware zeichnet sich dagegen nur durch Kostenfreiheit aus und muss die Punkte, wie sie bei freie Software gefordert sind, nicht erfüllen.

Freie Software hat jedoch in der Regel keinen Preis mehr, sobald sie einmal veröffentlicht ist. Viele, die an die geldbasierte Gesellschaft gewöhnt sind, sind zunächst einmal skeptisch gegenüber dieser Tauschfreiheit. Sie erwarten, dass Güter, für deren Erhalt sie nichts oder unverhältnismäßig wenig geben müssen, entweder Teil der Werbung sind oder sonst einen Pferdefuß haben. Freie Software ist aber weder Werbung noch hat sie sonst einen Pferdefuß. Insbesondere doppelt freie Software ist vielmehr von Anfang bis Ende jenseits des Tauschprinzips angesiedelt. Auch wenn die Teilnahme an einem freien Software-Projekt Geben und Nehmen beinhaltet, so ist der Erhalt von Leistungen jedoch nicht an die Erbringung von Leistungen gekoppelt. Tatsächlich werden die allermeisten NutzerInnen freier Software wenig oder gar nichts zu deren Weiterentwicklung leisten, können sie aber dennoch völlig uneingeschränkt nutzen.

Auf Grund der Konkurrenz sind Betriebsgeheimnisse in der Warenproduktion unerlässlich. Bei freier Software liegen dagegen die Quellen offen vor, sodass es gar keine Geheimnisse geben kann. Alle Interessierten können jederzeit das gesamte Know-how verwenden, das in einer Software enthalten ist.

Gleichzeitig lädt die Offenheit die NutzerInnen ein, die Software zu benutzen und Fehler und Wünsche zu melden, und sie lädt EntwicklerInnen ein, Verbesserungen und Erweiterungen einzubringen. Jeder auch noch so kleine Beitrag bringt alle voran. Freie Software saugt Kreativität und Wissen an. So herrscht Überfluss nicht nur beim Nehmen, sondern auch die Hineingabe ist potenziell unbegrenzt. Freie Software lädt zur *Kooperation* ein, sie funktioniert nach einem *Inklusionsmodell*.

Konkurrenz, also Durchsetzung auf Kosten anderer, wie wir sie zwischen Warenproduzenten erleben, gibt es insbesondere bei doppelt freier Software nicht. Wo es für eine bestimmte Problemstellung mehrere Programme gibt, so beziehen sie sich nicht konkurrenzförmig, also negativ, aufeinander. Entweder existieren die Projekte ohne besondere Beziehung nebeneinander oder es gibt eine mehr oder weniger starke Kooperation zwischen den Projekten. Proprietäre Software muss dagegen nicht nur das Nehmen begrenzen, sondern auch die Hineingabe ist beschränkt, denn nur ausgewählte EntwicklerInnen dürfen in den Quelltext sehen. Unsichere Software ist oft die Folge. Proprietäre Software basiert auf einem *Exklusionsmodell*.

Nun hat es in der Vergangenheit immer wieder Produktionsformen gegeben, die nicht vom Warenmarkt ausgegangen sind. Nicht selten sind Produkte zunächst im Hobbybereich ersonnen worden, und die Wirtschaft hat diese Erfindungen aufgegriffen. In solchen Fällen ist dem Hobbybereich bestenfalls eine Nische geblieben.

Anders bei freier Software. Wurde Software in der Frühzeit der Computer nicht als eigenständige Ware begriffen, so hatte sich in den 80er Jahren des vergangenen Jahrhunderts ein Warenmarkt für Software etabliert. Freie Software, die auf einem anderen Produktivkraftmodell als dem der Warenproduktion basiert, trat nun aus Sicht der NutzerInnen in direkte Konkurrenz zur proprietären Software. Im Gegensatz zu allen früheren Beispielen konnte sich freie Software aber nicht nur eine Nische sichern, sondern wächst im Gegenteil immer weiter und wird nach und nach zu einer ernsten Bedrohung für die proprietäre Software. Das neue Produktivkraftmodell, das wir in freier Software erkennen können, hat das Zeug dazu, die etablierte Warenwirtschaft zu ersetzen.

Freie Software ist nicht zuletzt im *Überfluss* vorhanden. Allein diese Eigenschaft ist ein nachhaltiges Hindernis, freie Software zu einer Ware zu machen. Um den Zusammenhang zwischen Produktion, Konsumption und Gesellschaft übergreifender zu verstehen, lohnt sich eine differenziertere Betrachtung der Begriffe Vorkommen, Begrenztheit und Knappheit.

Mit *Vorkommen* meinen wir, dass ein Gut vorkommt, unabhängig davon, ob wir es brauchen oder nicht. Die gesellschaftliche Dimension ist in diesem Begriff also nicht enthalten. Vorkommen kennt ein absolutes Maß, das z. B. im Begriff des Rohstoffvorkommens gefasst ist. Verleiht man dem Begriff ein zeitliches Maß, so ist er auch auf hergestellte Güter, also Produkte, übertragbar und er bezeichnet die zu einem bestimmten Zeitpunkt existierenden Produkte.

Mit *Begrenztheit* bezeichnen wir das Verhältnis zwischen der Verfügbarkeit eines Gutes und den Bedürfnissen der Menschen, dieses zu erhalten und zu nutzen. Gemessen am Bedarf, kann ein Gut in zu geringer, eben begrenzter Menge zur Verfügung stehen. Solche Begrenzungen können durch gesellschaftliches Handeln abgestellt werden, indem im einfachsten Fall vom begrenzten Gut mehr hergestellt wird. Produktion im allgemeinen Sinne bedeutet immer, gesellschaftlich mit Begrenzungen umzugehen.

Eine besondere Form des Umgangs mit Begrenzungen ist die Warenproduktion. Eine Ware darf nicht frei verfügbar sein, sonst ist sie keine, sie muss knapp sein. *Knappheit* ist eine geschaffene, soziale Form der Warenproduktion. Sie ignoriert wirkliche Begrenzungen und Vorkommen, um daraus die real wirksame Form Knappheit zu machen. Die soziale Form Knappheit produziert die Paradoxie des Mangels im Überfluss. Da abgelöst vom wirklichen Vorkommen, kann sie auch nicht nachhaltig sein.

6. Das Maintainer-Modell

Neben der Wertfreiheit spielt auch die Organisationsform eine große Rolle. Wer länger in der Softwareentwicklung tätig ist, weiß, dass bei Software-Projekten der soziale Prozess bezüglich der Organisation eine wesentliche Rolle spielt. Ist der soziale Prozess schlecht oder gar nicht organisiert, versinkt das interessanteste Projekt im Chaos, und Kreativität und Produktivität der Aktiven werden nachhaltig gestört. Wie ist dieser Prozess in freien Software-Projekten – genauer: in Projekten doppelt freier Software – organisiert?

Der entscheidende Unterschied zwischen der Entwicklung doppelt freier Software und anderer Software-Entwicklung besteht darin, dass alle ProjektteilnehmerInnen ausschließlich auf der Basis von *Freiwilligkeit* am Projekt teilnehmen. Da die Aktiven durch keinerlei entfremdete Anreize, wie zum Beispiel Entlohnung, an das Projekt gebunden sind, können sie das Projekt auch genauso freiwillig wieder verlassen.

So wie die Teilnahme nicht durch äußerliche Aspekte des Projektes bestimmt ist, ist es auch das Projekt insgesamt nicht. Vielmehr kann und muss sich jedes Projekt selbst Ziele setzen, sich *selbst organisieren*. Die Ziele beziehen sich dabei ausschließlich auf das gemeinsame Produkt und dessen Qualität.

Die Bedingungen der Freiwilligkeit der TeilnehmerInnen und der Selbstorganisation des Projekts bilden damit den Rahmen, in dem sich jede Organisation eines doppelt freien Software-Projekts abspielen muss. Wie dieser Rahmen in der Praxis gefüllt wird, ist nicht festgelegt. In sehr vielen Projekten gibt es jedoch das Maintainer-Modell.

Das Maintainer-Modell unterscheidet im Wesentlichen zwei Rollen, den oder die MaintainerIn und andere TeilnehmerInnen. Die Aufgaben der MaintainerIn bestehen im Wesentlichen darin, das Projekt generell auf Kurs zu halten. Die MaintainerIn entscheidet verbindlich über die grundsätzliche Richtung, in die die Software des Projekts weiterentwickelt werden soll, kümmert sich um die Einhaltung projektinterner Standards und darum, dass das Projekt sich bei Bedarf überhaupt weiterentwickelt. Nicht selten regelt die MaintainerIn auch die Außenkontakte für das Projekt. MaintainerInnen kommandieren jedoch nicht die anderen TeilnehmerInnen, vielmehr leisten diese freiwillig Beiträge zum Projekt in Form von Code, Dokumentation, Bug-Reports und vielem anderen mehr (Vgl. Raymond 2000).

Die speziellen Rahmenbedingungen doppelt freier Software führen zu Strukturen, die sich wesentlich von den bekannten Leitungsformen bei herkömmlicher Software–Entwicklung unterscheiden. Da die TeilnehmerInnen freiwillig am Projekt teilnehmen, können Entscheidungen nur getroffen werden, wenn der *Konsens* der wichtigen TeilnehmerInnen erreicht wird. Konsens meint hier nicht, dass alle zustimmen müssen (Einstimmigkeit), sondern Konsens ist vielmehr erreicht, wenn die TeilnehmerInnen einer Entscheidung nicht widersprechen müssen. Abstimmungen, wie sie in einigen Projekten vorgesehen sind, sind meist nur Mittel, um ein Stimmungsbild zu erzeugen.

Schafft es eine MaintainerIn in wichtigen Fragen nicht, einen Konsens herbeizuführen, so wird sie bald ohne TeilnehmerInnen da stehen. Gleichzeitig sind die TeilnehmerInnen darauf angewiesen, dass es Personen gibt, die die Aufgaben der MaintainerIn übernehmen und den Konsens organisieren. So ergibt sich eine gegenseitige Abhängigkeit zwischen MaintainerIn und anderen TeilnehmerInnen. Für das Produktivkraftmodell, das wir bei freier Software beobachten können, sind konsensorientierte Organisationsformen, wie z. B. das Maintainer-Modell, die Konflikte optimal ausbalancieren, unabdingbar.

7. Eine neue Qualität von Produktivkraftentwicklung

Eine wichtige Basis der Argumentation im Projekt Oekonux ist, dass es sich bei dem Phänomen freie Software um ein Beispiel für ein qualitativ neues Modell von *Produktivkraftentwicklung* handelt. Unter Produktivkraftentwicklung verstehen wir die historische Entwicklung der *Produktivkraft*, wobei Produktion in diesem Zusammenhang als das Stoffwechselverhältnis zwischen Mensch und (äußerer) Natur betrachtet wird. Produktivkraft fasst das Verhältnis zwischen Menschen, Produktionsmitteln und der Natur zusammen (Marx 1974).

Wir können drei Dimensionen von Produktivkraft unterscheiden. Die Dimension des *Inhalts* beschreibt, was der Inhalt menschlicher Tätigkeit ist – also die Art der Produkte, der Bezug zur Natur und die verwendeten Produktionsmittel. Im hier betrachteten Fall also freie Software und die für ihre Herstellung verwendeten Produk-

tionsmittel. Die Dimension der *Form* beschreibt die Art und Weise der Organisation des Produktionsprozesses – ob also z. B. Arbeitsteilung eingesetzt wird. Bei freier Software gehört die Selbstorganisation und das Maintainer-Modell in diesen Bereich. Die Dimension der *Produktivität* beschreibt die produzierte Gütermenge pro Zeiteinheit.

Wenn sich diese Dimensionen der Produktivkraft verändern, sprechen wir von einem qualitativen Schritt in der Produktivkraftentwicklung. Bei freier Software sehen wir eine Veränderung vor allem beim Inhalt und der Form der Produktivkraft. Im Folgenden werden einige Aspekte freier Software beschrieben, die wir für Hinweise auf diese neue Qualität halten.

Software insgesamt ist eine Produktgruppe, die erst seit einigen Jahrzehnten existiert. Ihre Nutzung setzt das Vorhandensein von Computern, mithin also hochmoderner Geräte voraus. Software, damit auch freie Software, ist also ein hochmodernes Produkt, das auf einem früheren Stand von Produktivkraftentwicklung gar keinen Sinn gemacht hätte. Freie Software befindet sich als Produkt an der *Spitze der allgemeinen Produktivkraftentwicklung*.

Nach wie vor unterliegen die Paradigmen, unter denen Software hergestellt wird, einem schnellen Wandel: Strukturierte Programmierung, Objektorientierung, Wasserfallmodell und agile Methoden – um nur ein paar zu nennen – haben sich innerhalb weniger Jahre abgelöst. Wir erleben ein Entwicklungstempo an den Wurzeln einer Technologie, das in anderen Ingenieursdisziplinen längst Vergangenheit ist. Mit einigem Recht kann freie Software als Produktionsweise ebenfalls als neues Paradigma bezeichnet werden. Einer der fundamentalen Unterschiede wird schon in der berühmt gewordenen Tanenbaum-/Torvalds-Debatte aus der Frühzeit der Linux-Entwicklung deutlich. Der Informatik-Professor Tanenbaum bezeichnete darin das von Torvalds für Linux avisierte Entwicklungsmodell als nicht praktikabel, da es nicht möglich sei, tausend Primadonnen zu kontrollieren. Die lakonische Antwort von Torvalds, die das Maintainer-Modell im Wesentlichen vorwegnimmt, bestand darin, dass es nicht seine Absicht sei, zu kontrollieren.[6] Diese Aufgabe von Kontrolle bezieht sich dabei sowohl auf die „Primadonnen" als auch auf den Code selbst. Freie Software gehört hinsichtlich seiner Produktionsweise zu einem der *innovativsten Ansätze*.

Freie Software wird nicht nur auf Computern benutzt und über das Internet verteilt, sondern auch mit Hilfe von Computern und Internet entwickelt. Computer allgemein und speziell ihre Anwendung in Form des Internets sind die zentralen Produktionsmittel für die Entwicklung freier Software. Diese Produktionsmittel gehören ebenfalls zu den *am weitesten entwickelten Produktionsmitteln*, die die Menschheit bisher hervorgebracht hat.

Im Gegensatz zu Produktionsmitteln vorangegangener Produktivkraftepochen sind Computer und das Internet auf Grund ihrer Universalität nicht auf Produktion digitaler Güter festgelegt, sondern können auch zum Spielen, zum Musikmachen, zum Diskutieren etc. eingesetzt werden. Die Produktionsmittel freier Software sind zunehmend *Teil der allgemeinen Infrastruktur* der sich am Horizont abzeichnenden Informationsgesellschaft.

6 Siehe hierfür http://www.educ.umu.se/~bjorn/mhonarc-files/obsolete/msg00089.html oder Torvalds (2001).

Diese allgemeine Infrastruktur ist heutzutage so preiswert und gleichzeitig allgemein nützlich geworden, dass sie in den hochindustrialisierten Regionen bereits beinahe überall verfügbar ist. Die Produktionsmittel, auf denen freie Software beruht, befinden sich also in *breiter privater Verfügung*. Auch dies ist ein Aspekt, der für Produktionsmittel vorangegangener Produktivkraftepochen nicht gilt.

Die Teilnahme an freien Software-Projekten ist nicht an Staaten oder Kulturkreise gebunden. Ganz selbstverständlich finden sich alle Interessierten an einem Projekt via Internet und kooperieren, um ein Produkt zu erstellen, das ihnen entspricht. Entwicklung freier Software ist *transnational*. Sie bezieht sich hinsichtlich ihrer Lizenzen auf die nationalstaatlichen Rechtssysteme der früheren Produktivkraftepoche und definiert somit einen eigenen Raum jenseits der Nationalstaaten.

Bemerkenswert ist auch, dass das gesamte Phänomen freier Software aus der Zivilgesellschaft kommt. Weder staatliche Agenturen noch Firmen haben freie Software hervorgebracht. Erst in neuerer Zeit, nachdem freie Software bereits erhebliche Erfolge erzielt hat, beginnen staatliche Einrichtungen und die Wirtschaft, auf den fahrenden Zug aufzuspringen. Freie Software würde sich auch unabhängig von diesen Einflüssen weiter entwickeln. Anstatt, dass Staat oder Wirtschaft die Kontrolle übernehmen, gibt es viele Beispiele dafür, dass sie sich den Gegebenheiten freier Software anpassen. So hat beispielsweise IBM zu Beginn seines Engagements im Bereich freier Software explizit darauf hingewiesen, dass man als großer Player behutsam mit der Community umgehen müsse – was allem Anschein nach bis heute erfolgreich durchgehalten wird und zu einer gewissen Reputation in der Community geführt hat. IBM hat verstanden, dass der Versuch eine Einschränkung der Freiheit die Kuh schlachten würde, die sie gerne melken möchte. So haben Firmen wie IBM ein großes Interesse daran, freie Software-Projekte zu unterstützen, da sie von den Leistungen der Community in höherem Maße profitieren, als sie es durch eine Kontrolle des Code erreichen würden.

8. Digitale Kopie als technologische Grundlage

Wir haben erläutert, dass bei der Entwicklung freier Software die Selbstentfaltung den individuell-sozialen Aspekt der Produktivkraftentwicklung bildet. Als technologische Seite dieser Entwicklung tritt die *digitale Kopie* hinzu. Während der Aspekt der Selbstentfaltung als gesellschaftliche Potenz schon immer da gewesen ist, handelt es sich bei der digitalen Kopie um eine historisch neue Potenz. Erst durch diesen technologischen Fortschritt ist die Ausdehnung der Selbstentfaltung als Grundlage eines Produktivkraftmodells möglich geworden.

Die digitale Kopie, die Möglichkeit also, digitale Informationen zu reproduzieren, hat im technologischen Sinne einige Eigenschaften, die sie älteren Technologien gegenüber voraus hat.

Während analoge Reproduktionen von Information immer mit Verfälschungen zu kämpfen haben, liefert die digitale Kopie eine *exakte Reproduktion* des Originals: Original und Kopie sind nicht zu unterscheiden. Mit diesem technologischen Fortschritt werden Begrenzungen der Verfügbarkeit digitaler Informationen nachhaltig beseitigt.

Zwei weitere Tatsachen moderner Technologieentwicklung geben der digitalen Kopie aber erst richtig Sprengkraft. Einerseits ist diese Reproduktionstechnik nämlich mittels Computern für sehr viele Menschen täglich und selbstverständlich verfügbar. Diese *breite Verfügbarkeit* führt dazu, dass die digitale Kopie kaum noch Einschränkungen unterliegt. Digital-Rights-Management-Technologien sind so gesehen nichts anderes als der (krampfhafte) Versuch, diese basale Eigenschaft moderner Technologie wieder zurückzunehmen, denn tatsächlich sind die beiden fundamentalen Operationen von Computern die Manipulation und die Kopie von digitalen Daten.

Andererseits steht mit dem Internet, das nichts anderes als eine planetenumspannende Fernkopiereinrichtung ist, eine Einrichtung zur Verfügung, die es ermöglicht, dass Informationsgüter auf einfachste Weise *global* verfügbar gemacht werden können. Das Internet verbindet die individuelle Verfügung über Informationsgüter mit dem allgemeinen Zugang zu ihnen.

Ein weiterer, eher subtiler Aspekt digitaler Kopie ist ihre *Universalität*: Der Inhalt, die Bedeutung des zu kopierenden Informationsguts, ist für den Vorgang der digitalen Kopie völlig unerheblich. Texte, Bilder, Musik, Programme können mit der gleichen Technologie reproduziert werden, sobald sie als Bytestrom vorliegen. So, wie die Kraftmaschinen der industriellen Ära (Dampfmaschine, vor allem aber Elektromotor) eine Basistechnologie für beliebige Anwendungen mechanischer Kraft und damit für die Industriegesellschaft bilden, so bildet die digitale Kopie eine Basistechnologie für die Informationsgesellschaft.

Auf dieser Grundlage ist das Internet von Beginn an auch als Kommunikationsmittel genutzt worden. Wie keine Kommunikationseinrichtung zuvor ermöglicht das Internet *globale Kommunikation* in Echtzeit. Es ist jetzt möglich, dass Menschen mit gleichen Bedürfnissen unabhängig von ihrem Standort in dem Tempo kommunizieren, das ihnen und ihrer Tätigkeit angemessen ist. Diese Kooperationsmöglichkeit ist wie die digitale Kopie selbst eine unabdingbare Voraussetzung für die Entfaltung freier Software.

9. Freie Software als Keimform

Eine der zentralen und nicht unumstrittenen Thesen im Projekt Oekonux ist die These von der freien Software als *Keimform einer neuen Gesellschaft* (Vgl. Kurz 1997). Unter einer Keimform verstehen wir ein Phänomen, das in den Rahmen eines bestehenden Gesamtsystems eingebettet ist, gleichzeitig aber Eigenschaften hat, die über die Logik des umgebenden Gesamtsystems hinausgehen und eine mögliche, neue Entwicklungsrichtung des Gesamtsystems darstellen können. Eine Keimform ist dabei noch keine vollständig entfaltete Form einer Gesellschaft, sondern zeigt nur einige identifizierbare Merkmale.

Eine neue Form entfaltet sich dagegen in mehreren Schritten. Das so genannte *Fünfschritt-Modell*, aus dem der Begriff Keimform stammt, erfasst in allgemeiner Weise, wie es innerhalb von Entwicklungsprozessen zu qualitativen Übergängen kommt. Es erklärt, wie Neues entsteht und sich schließlich durchsetzen kann. Das Fünfschritt–

Modell kommt ursprünglich aus der Kritischen Psychologie (Holzkamp 1983), aus der Analyse qualitativer Entwicklungsschritte in der Evolution. Die fünf Schritte sind:

1. Entstehung der Keimform
   Alles, was es selbstverständlich und allgegenwärtig gibt, ist irgendwann einmal etwas Neues, ganz und gar nicht Selbstverständliches gewesen. Über mehrere Schritte hat sich das Neue schließlich durchgesetzt. Dieses Neue, das später einmal Altes sein wird, nennt man Keimform. Keimformen können in Nischen und Sonderbereichen entstehen. Sie leben vom und im Alten, besitzen aber schon Formen des Neuen.

2. Krise der alten Form
   Keimformen erlangen nur Bedeutung, wenn das Alte in die Krise gerät. Das Alte kann im Wesentlichen aus zwei Gründen in die Krise geraten. Zum einen können sich äußere Bedingungen so dramatisch oder so schnell verändern, dass das alte Prinzip darauf nicht mehr angemessen reagieren kann. Zum anderen kann sich das Alte selbst erschöpft haben, wenn alle Entwicklungspotenzen ausgereizt sind. Stagnation wäre eine Reaktionsform, Zerfall eine andere.

3. Keimform wird zur wichtigen Entwicklungsdimension
   Unter den Bedingungen der Krise des Alten kann die Keimform die Nischen verlassen und sich quantitativ ausbreiten. Sie wird zu einer wichtigen Entwicklungsdimension innerhalb der noch dominanten alten Form. Diese Etablierung der Keimform kann zwei Richtungen einschlagen: Sie führt zur Integration in das Alte und zur Übernahme der alten Prinzipien oder die Keimform behauptet sich auf Grund der neuen Prinzipien immer besser im und neben dem Alten. Im ersten Fall geht der Keimformcharakter verloren, im zweiten Fall wird das Neue gestärkt. Das Alte kann in beiden Fällen von einer integrierten oder gestärkten Keimform profitieren und Krisenerscheinungen abmildern.

4. Keimform wird zur dominanten Größe
   Die frühere minoritäre Keimform wird zur dominanten Form der Entwicklung. Das Neue setzt sich durch, weil es hinsichtlich einer wichtigen Dimension des Gesamtprozesses besser ist. Damit endet der Keimformcharakter des Neuen. Nun sind seine Prinzipien bestimmend und verdrängen nach und nach oder auch schlagartig die überkommenen, nicht mehr funktionalen Prinzipien des Alten. Das Neue wird das selbstverständliche Allgegenwärtige.

5. Umstrukturierung des Gesamtprozesses
   Schließlich strukturieren sich alle Aspekte des Gesamtprozesses in Bezug auf das bestimmende, jetzt selbstverständliche Neue hin um. Das betrifft vor allem auch solche Prozesse, die im Gesamtprozess nicht bestimmend, sondern nur abgeleitet sind. Mit diesem Schritt ist nun potenziell wieder der erste Schritt eines neuen Fünfschrittes erreicht: Keimformen können auftreten, das dann alte Neue gerät in die Krise usw.

Alle Phasen können über kürzere oder längere Zeiträume andauern und es kann jederzeit Rückschritte geben. Nichts ist vorgegeben oder determiniert. Vollständig begriffen kann ein Fünfschritt der Entwicklung erst werden, wenn er vollzogen wurde und erst im Nachhinein kann man die frühere Keimform sicher identifizieren. Mitten im Entwicklungsprozess begriffen kann das Fünfschrittmodell helfen, die Sinne zu schärfen, um handlungsfähiger zu werden. Die umstrittene These im Projekt Oekonux lautet nun: „Bei freier Software haben wir es mit einer Keimform einer neuen Gesellschaft zu tun".

Wie beschrieben, zeichnet sich freie Software durch Wertfreiheit, Selbstentfaltung, Selbstorganisation und Globalität aus. Das alte Prinzip der Warengesellschaft basiert demgegenüber auf dem Wertgesetz, der Selbstverwertung, der Entfremdung und den Nationalstaaten. Die alte Form, die Warengesellschaft, ist erkennbar in der Krise. So versprach noch in den siebziger Jahren des vergangenen Jahrhunderts eine weitere Entfaltung und Vertiefung der Warengesellschaft nicht nur den Menschen der hoch industrialisierten Staaten individuelle Wohlstandsmehrung, sondern dieses Versprechen konnte auch erkennbar eingehalten werden. Von einem Fortschritt in dieser Hinsicht spricht heute niemand mehr. Vielmehr wird im Rahmen von Arbeitslosigkeit und Sozialabbau ganz offen und in steigendem Tempo nur noch darüber diskutiert, wie die Senkung des Lebensstandards breiter Bevölkerungskreise auch der hoch industrialisierten Staaten weiter vorangetrieben werden soll. Demgegenüber hat sich freie Software als neue Form von Reichtum etabliert, das jenseits der Formen der Verwertung existiert. Umstritten ist, ob freie Software bereits eine wichtige Entwicklungsdimension innerhalb der alten Form (dritter Schritt) geworden ist oder sich noch in einer der früheren Phasen befindet.

10. GPL-Gesellschaft

In der Diskussion um die gesellschaftlichen Potenzen des Entwicklungsmodells, das sich nach der These in freier Software keimförmig zeigt, kam der Begriff der GPL–Gesellschaft auf (Merten 2000). Er bezeichnet eine mögliche zukünftige Gesellschaftsform, die auf den Prinzipien der Entwicklung freier Software beruht. Es ist aus verschiedenen Gründen nicht seriös möglich, ein detailliertes Bild einer solchen Vorstellung zu entwerfen. Entlang der Prinzipien der Entwicklung freier Software lassen sich aber einige Rahmenelemente feststellen.

Ein wichtiges Element der Entwicklung freier Software ist die Tatsache, dass die verwendeten Produktionsmittel vergleichsweise vielen Menschen Selbstentfaltung ermöglichen. Der innere Grund dafür ist, dass Computer als universelle, Information verarbeitende und programmierbare Maschinen *unendlich viele Freiheitsgrade* haben. Diese Freiheitsgrade können von Menschen zur Entfaltung ihrer individuellen Kreativität genutzt werden.

In einer GPL-Gesellschaft wäre diese Eigenschaft tendenziell auf alle Produktionsmittel übertragbar. Dies bedeutet, dass der Maschinenpark, den die Industriegesellschaft für die Nutzung unter den entfremdeten Bedingungen der Lohnarbeit hervorgebracht hat, umgearbeitet oder neu entworfen werden muss: Die Arbeit an

einem Fließband dürfte beispielsweise nur für die allerwenigsten Menschen zur Selbstentfaltung führen, weswegen sie endgültig verschwinden müsste.[7]

Auch die weitere und noch beschleunigte *Automatisierung von Arbeitsprozessen* ist ein Mittel, um maximale Selbstentfaltung zu gewährleisten. Arbeitsprozesse, die im Kern von Maschinen übernommen werden, müssen nicht mehr von Menschen erledigt werden. Sie können sich den Aufgaben zuwenden, die genuin menschliche Fähigkeiten erfordern und nicht von Maschinen übernommen werden können.

Wie wenige andere Beispiele macht freie Software sichtbar, dass Selbstentfaltung nicht sinn- und zweckfreies Tun sein muss, wie es uns die Freizeitindustrie weismachen will. Vielmehr ist das Ergebnis der Entwicklung freier Software ein Produkt, das für viele Menschen nützlich ist. Selbstentfaltung, wie sie in freier Software praktiziert wird, hat also nicht nur für das Individuum eine positive Funktion, sondern nutzt der gesamten Gesellschaft. In einer GPL-Gesellschaft hätten noch sehr viel mehr Tätigkeiten diesen Charakter der *unmittelbaren Verknüpfung von individuellem und gesellschaftlichem Nutzen*.

Nicht zu vergessen ist, dass existierende freie Software frei verfügbar ist. Diese Eigenschaft ist einerseits eine Folge des offenen Entwicklungsprinzips, das die maximale Inklusion aller Interessierten zum Ziel hat. Unter den Bedingungen der universellen digitalen Kopierbarkeit führt dies zusammen mit dem *Copyleft* tendenziell zu allgemeiner freier Verfügbarkeit der Produkte. Andererseits ist diese freie Verfügbarkeit auch Voraussetzung für die blühende freie Software-Landschaft, denn auch die EntwicklerInnen von freier Software setzen auf von anderen entwickelter freier Software auf.

Die freie Verfügbarkeit ist also sowohl Folge als auch Voraussetzung des gesamten Entwicklungsmodells. Diese enge Verschränkung wäre in einer GPL-Gesellschaft ausgedehnt auf alle Informationsgüter sowie auf materielle Güter.

In einer GPL-Gesellschaft wären folglich viele Einrichtungen der Arbeitsgesellschaft überflüssig. Wo Güter frei verfügbar sind, ist die Form der Ware nicht mehr zu halten, die davon lebt, dass Güter künstlich verknappt werden. Werden keine Waren mehr – wohl aber Güter – produziert, so ist auch kein Geld mehr notwendig, das die Vergleichbarkeit von Waren vermittelt: Wo Güter frei zur Verfügung stehen, ist der Tausch eines Guts gegen ein anderes zur überflüssigen Handlung geworden. Nicht zuletzt würde unter den Bedingungen der GPL-Gesellschaft Entfremdungspotential an vielen Stellen tendenziell abgeschafft. Die wichtigste Produktivkraft einer GPL-Gesellschaft wäre die menschliche Selbstentfaltung.

In einer GPL-Gesellschaft würde die Selbstentfaltung der Individuen zur unmittelbaren Voraussetzung für die Selbstentfaltung aller: Nur wenn sich die Individuen entfalten können, entstehen Produkte, die für alle nützlich sind. Gleichzeitig sind diese nützlichen Produkte und deren freie Verfügbarkeit die Grundlage für die individuelle Tätigkeit: Die Selbstentfaltung aller ist also auch die Voraussetzung für die Selbst-

7 Bereits in der Industriegesellschaft gibt es verschiedene Versuche, die Selbstentfaltung in die Produktion zu integrieren und so die Kreativitätsreserven der Lohnabhängigen zu mobilisieren. Allerdings können diese Versuche nicht die strukturelle Schranke der Entfremdung des Arbeitsprozesses überwinden (Vgl. auch Gorz 2004).

entfaltung der Individuen. Wir haben es mit einem sich selbst verstärkenden Prozess zu tun, der die langfristige Tragfähigkeit einer GPL-Gesellschaft in einem günstigen Licht erscheinen lässt.

11. Historische Schritte im Vergleich

Wenn wir davon ausgehen, dass freie Software ein Hinweis auf einen fundamentalen Schritt in der Produktivkraftentwicklung ist, so kann ein Vergleich mit dem letzten fundamentalen Schritt in der Produktivkraftentwicklung interessante Einsichten geben. Die letzte fundamentale Änderung war der Umbruch von den feudal geprägten Gesellschaften des Spätmittelalters zu den industriell geprägten Gesellschaften der Neuzeit mit Beginn der Aufklärung. Die gerade erfundenen Industriemaschinen erforderten für ihren Betrieb hinsichtlich Technik und Organisation von Menschen eine völlig neue Produktionsweise. Dies verwob sich mit sozialen und ideologischen Entwicklungen, wie beispielsweise der Idee der Nationalstaaten, sodass innerhalb eines historisch relativ kurzen Zeitraums diese Entwicklung der Produktivkräfte zu einer tiefgreifenden Umstrukturierung der Gesellschaft führte.

Einerseits wurden damals menschliche, tierische und einige wenige natürliche Kraftquellen (Wasser, Wind) durch moderne Kraftquellen (Dampf, Elektrizität) abgelöst. Andererseits wanderte das Know-how über die Arbeitsprozesse von den sie ausführenden Menschen in die mechanische Konstruktion der Maschinen, so dass menschliche Arbeitskraft nur noch als Ergänzung zu den Maschinen benötigt wurde. Das Ergebnis dieser Umstellung waren eine Fülle neuer, nützlicher Produkte sowie Großprojekte, die ohne Einsatz industrieller Methoden nicht denkbar gewesen wären. Eine unrühmliche Antriebsfeder war die industrielle Entwicklung von Militärtechnik.

Gleichzeitig strukturierte sich die Lebensweise der Menschen tiefgreifend um. Die Art und Weise, wie Menschen in der feudalen, subsistenzorientierten Produktionsweise ihr Leben organisiert haben, war für die industrielle Produktionsweise in vielerlei Hinsicht ungeeignet. Um nur ein Beispiel herauszugreifen: Der vorindustrielle Umgang mit Zeit war weitgehend an den Hell-/Dunkelphasen und den konkreten, unmittelbaren eigenen Notwendigkeiten orientiert. Dies ging so weit, dass in manchen Klöstern die Länge einer Stunde über den Jahreslauf variierte. Für die industrielle Produktion, bei der der Zeittakt durch die Maschinen vorgegeben wird, waren das völlig unbrauchbare Verhältnisse. Es soll nicht verschwiegen werden, dass diese Umstrukturierung der Lebensweise durchaus nicht immer freiwillig geschah, sondern in erheblichem Ausmaß auch mit dem Einsatz von Gewalt einherging.

Auch beim Übergang von der feudalen zur bürgerlichen Gesellschaftsform lassen sich Keimformen ausmachen. So kann beispielsweise das Handelskapital, das sich bereits im Frühmittelalter auszubilden begann, als frühe Keimform betrachtet werden, bei der sich der kapitalistische Umgang mit Geld entwickelte. Die frühindustrielle Textilindustrie war für diesen Schritt in der Produktivkraftentwicklung die Keimform, in der schon viele Formen der späteren Industriearbeit auftauchten. Insbesondere traten hier sowohl die standardisierte Massenproduktion als auch der massenhafte Einsatz von bezahlten Arbeitskräften auf.

Es lässt sich an diesem Beispiel auch sehr schön betrachten, wie die feudalen Strukturen, die alten dominanten Strukturen also, sich an diese Entwicklung anpassten und sie auch nutzten. Die gesamte Kriegsproduktion, die die Fürsten für die Führung ihrer Kriege brauchten, hatte sich von der subsistenzorientierten Produktion über die Jahrhunderte hin vollständig entbettet. Die Kriegsproduktion der späten Feudalherren, in der auch industrielle Formen relativ früh auftauchten, war nur mit Geld überhaupt zu bewerkstelligen. Das dafür notwendige Steuersystem trieb das Geldwesen weiter an, das sich später als eine entscheidende Grundlage nach-feudaler Gesellschaftsformen herausstellen sollte. Söldnerwesen sowie stehende Heere können als frühe Form bezahlter „Arbeitskraft" betrachtet werden.

Heute wissen wir, dass diese Entwicklung, die in frühen Keimformen bereits erkennbar war, zu einer grundlegenden Umwandlung der Gesellschaftsform geführt hat. Auch wenn die Fürsten die Aufklärung und die mit ihr verbundende, industrielle Produktionsweise teilweise begrüßten oder sogar vorantrieben, so stellte sich doch heraus, dass die feudale Produktionsweise mit ihrem Privilegiensystem und der Leibeigenschaft industrieller Produktion unangemessen war. Die industrielle Produktionsweise musste sich also neue Grundlagen schaffen, die durch Geldwirtschaft und freie Lohnarbeit gekennzeichnet waren. Auch die gesamte gesellschaftliche Organisation folgte nach und nach dieser Entwicklung der Produktivkraft – am sichtbarsten in der Gründung bürgerlicher Nationalstaaten.

Heute sind wir an einem Punkt angekommen, wo sich die mechanische Konstruktion von Maschinen immer weiter von einem konkreten Arbeitszweck ablöst. Industrieroboter, Fabber[8] etc. sind nicht durch ihre Konstruktion auf einen bestimmten Arbeitsprozess festgelegt, sondern ihre mechanische Konstruktion steckt nur noch den Rahmen ihrer Möglichkeiten ab. Die Produktion konkreter Gegenstände ist bei diesen Maschinen bereits Gegenstand von Software, womit das Know-how über die Arbeitsprozesse zum Informationsgut wird. Computer verwenden diese Informationsgüter als Programme in automatisierten Prozessen, sodass menschliche Energie und Kreativität nur noch dafür benötigt wird, die Informationsgüter selbst zu erstellen.

Die Nutzung von Industriemaschinen erforderte auf Grund ihrer Beschränkungen eine Anpassung der Menschen an die Notwendigkeiten der Maschinerie, was in vielerlei Hinsicht letztlich eine Unterwerfung bedeutete. Die Produktion von Informationsgütern ist hingegen ein kreativer Prozess, bei dem gerade die schöpferischen Qualitäten des Menschen gefragt sind, die durch Unterwerfung vernichtet werden. Zog die beginnende Industriegesellschaft eine Unterwerfung der Menschen nach sich, so erfordert die beginnende Informationsgesellschaft eine Freisetzung der unbeschränkten Selbstentfaltung von Menschen.

Während sich beim Übergang von den feudalen zu den bürgerlichen Gesellschaften der Schwerpunkt der Produktion von der Nutzung des Bodens zur industriellen Produktion materieller Güter verlagerte, so verschiebt sich der Schwerpunkt der

8 Ein Fabber (*digital fabricator*) wird auch als „Fabrik in einer Kiste" bezeichnet. Fabber erzeugen dreidimensionale Werkstücke direkt aus digitalen Daten, z. B. aus Kunststoff oder Metall. Fabber werden derzeit hauptsächlich im „Rapid Prototyping" zur Fertigung von Prototypen und Modellen verwendet.

Produktion beim Übergang in die *heraufziehende Informationsgesellschaft* auf die Produktion neuer Informationsgüter. Der Wechsel zur Industrieproduktion erforderte einen fundamentalen Wechsel in der Gesellschaftsform. Informationsgüter, die in fast allen Aspekten anderen Bedingungen unterliegen als materielle Güter, erfordern eine ebensolche Umstrukturierung. Eine grundlegende Änderung der Gesellschaftsform erscheint unabdingbar.

Literaturverzeichnis

Gorz, A. (2004), *Wissen, Wert und Kapital. Zur Kritik der Wissensökonomie*, Rotpunktverlag, Zürich.

Himanen, P. (2001), *The Hacker Ethic and the Spirit of the Information Age*, 1. Aufl., Random House, Vintage, UK.

Holzkamp, K. (1983), *Grundlegung der Psychologie*, Campus Verlag, Frankfurt am Main.

Kurz, R. (1997), 'Antiökonomie und Antipolitik. Zur Reformulierung der sozialen Emanzipation nach dem Ende des „Marxismus"', *Krisis* **19**. http://www.giga.or.at/others/krisis/r-kurz_antioekonomie-und-antipolitik_krisis19_1997.html.

Marx, K. (1974), *Grundrisse der Kritik der politischen Ökonomie*, Dietz-Verlag, Berlin/DDR. Rohentwurf 1857–1858.

Meretz, S. (2000*a*), 'GNU/Linux ist nichts wert – und das ist gut so!', http://www.kritische-informatik.de/?lxwertl.htm.

Meretz, S. (2000*b*), *Linux Co. Freie Software – Ideen für eine andere Gesellschaft*, AG SPAK Verlag, Neu-Ulm.

Meretz, S. (2004), 'Freie Software. Über die Potenziale einer neuen Produktionsweise', *Widerspruch* **45**.

Merten, S. (2000), GNU/Linux – Meilenstein auf dem Weg in die GPL-Gesellschaft, *in* 'Dokumentation LinuxTag 2000'. http://www.oekonux.de/texte/meilenstein/.

Raymond, E. S. (2000), 'The Cathedral and the Bazaar', http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/.

Torvalds, L. (2001), *Just for Fun – Wie ein Freak die Computerwelt revolutionierte*, Carl Hanser Verlag.

Open Source – Die Rückkehr der Utopie?

CHRISTIAN F. GÖRLICH UND LUDGER HUMBERT

(CC-Lizenz siehe Seite 463)

> „You think you know when you learn, are more sure when you can write, even more when you can teach, but certain when you can program." (Perlis 1982)

Die hier getätigten Überlegungen wollen zunächst als Beitrag zur *Selbstaufklärung* der Open-Source-Bewegung verstanden werden. Unter Rückgriff auf eine differenzierende Begrifflichkeit und klassische Problemstellungen der Philosophie in ihrem besonderen Verhältnis zu den Einzelwissenschaften – hier insbesondere der Soziologie und Wissenschaftsgeschichte – sollen Fragehorizonte rekonstruiert werden, die Antworten erhoffen lassen, die dann in den Auseinandersetzungen um Open Source auch den Anspruch der Akteure auf die Mündigkeit zu stärken versprechen. Aus dem aufgewiesenen Fragehorizont wird in diesem Beitrag eine wissens- /wissenschaftssoziologische Problemstellung thematisiert, die Open Source als gesellschaftliches Subsystem in den Blick nimmt und im Anschluss an Merton (1972) über die leitenden Normen reflektiert. In einem Exkurs wird unter Bezug auf Hagstrom (1965) dem Ursprung der gegenwärtigen Diskussion um die Open-Source–Bewegung als einer Tauschgesellschaft nachgegangen. Abschließend wird mit Blick auf den Titel dieses Beitrages darüber räsoniert, welche Funktionen das Subsystem Open Source in Analogie zu ähnlichen Subsystemen, die sich um Kreativität formieren, für das gesellschaftliche Gesamtsystem wahrnehmen könnte. Es geht insbesondere auch darum, den Begriff einer „Informatischen Vernunft" zu etablieren und in seinem Verhältnis zur Allgemeinbildung zu bestimmen. Informatische Vernunft will nicht nur *instrumentelle Kenntnis* sein, Informatische Vernunft will in dem epochaltypischen Schlüsselbereich der „neuen technischen Steuerungs-, Informations- und Kommunikationsmedien" (Klafki 1991*a*, S. 59) den philosophischen Anspruch der Aufklärung wach halten.

1. Open Source – Fragestellungen im Kontext philosophischer Betrachtungen

Open Source als geistige Bewegung in einer sich zur Wissensgesellschaft wandelnden Welt philosophisch zu denken, eröffnet vielfältige Perspektiven und Fragestellungen. Die nachfolgende Aufzählung benennt einige Facetten:

- Open Source unter dem Blickwinkel des für jede Gesellschaft konstitutiven Eigentumsverständnisses (Vgl. Künzli 1986; Lutterbeck und Gehring 2004, Kap. 5, S. 331 ff.)
- Open Source unter dem Blickwinkel einer normengeleiteten Wissenschaftsentwicklung (ethischer und wissenschaftssoziologischer Zugang)
- Open Source unter dem Blickwinkel der Produktion: Arbeit und/oder Spiel (entfesselte Kreativität)
- Open Source unter dem Blickwinkel emanzipatorischer Entwicklungspotenziale

Der zuletzt genannte Blickwinkel benennt auch jenseits postmoderner Attitüden das eigene Erkenntnisinteresse. Wie in These 6 von Görlich und Humbert (2003, S. 90) ausgeführt, wird hier Philosophie nicht – wie noch bei Hegel – metaphysisch als Platzanweiserin[1] für die Einzelwissenschaften, sondern eher in Anlehnung an Habermas als Platzhalterin der Vernunft in den sich immer weiter ausdifferenzierenden und auch sich verheiratenden Einzelwissenschaften verstanden. Die Reflexion von Grundlagen und zentralen Begriffen und der nachdenkende Rückgriff auf klassische Problemstellungen fühlt sich dabei nach wie vor der Aufklärung verpflichtet. Unter „klassisch" verstehen wir in diesem Zusammenhang in Anlehnung an Luhmann, dass die in der Vergangenheit angesprochenen Probleme heute noch strukturell wirkmächtig sind, auch wenn wir sie heute in einer anderen Sprache oder bezüglich einer anderen (neuen) Materie reformulieren müssen. Und mit „Aufklärung" knüpfen wir an Kants bekannte Rede von der „Aufklärung als Ausgang aus der selbstverschuldeten Unmündigkeit" an. Angesichts des Patchwork-Charakters der Welt sollte dieses Festhalten an der Aufklärung nicht als naiv missverstanden werden, sondern eher ironisch im Sinne Richard Rortys. Ironisch meint hier eine ungläubige Haltung gegenüber jedem Anspruch auf etwas Letztgültiges. Rorty steht hier ganz in der Tradition von Nietzsches Behauptung, dass Wahrheiten nur Illusionen seien, „von denen man vergessen hat, dass sie welche sind." Sprache, Selbstbild, Kultur und Lebensform sind kontingent. Soweit ist Rorty Teil des postmodernen Diskurses, der Aufklärung als Blendwerk erscheinen lässt. Jedoch hier wendet sich Rorty vom postmodernen Diskurs ab. Rortys Leistung besteht darin, „einen postmodernen Standpunkt zu beziehen, ohne damit auch der politischen Resignation das Wort zu reden. Vielmehr bleibt Rorty ein leidenschaftlicher Verfechter der Aufklärung, hält allerdings deren *Vokabular*, das um den

1 Für personenbezogene Bezeichnungen in diesem Beitrag wird – abgesehen von Zitaten – für geschlechtsspezifische Bezeichnungen das generische Femininum gewählt. Männer mögen sich nicht ausgeschlossen fühlen.

Begriff einer universal gültigen Vernunft gruppiert ist, für „veraltet" – im Gegensatz zu Habermas. Rorty versteht sich als Vertreter einer bildenden und zugleich ironischen Philosophie: Bildend meint hier die Negation einer systematischen Philosophie und positiv: „ein Sich-bekannt-Machen mit anderen Kulturen, geschichtlichen Epochen, [. . .] ein fortwährendes Ins-Gespräch-Kommen mit Fremden, ein Neubefragen des Bekannten". (Breuer et al. 1996, S. 123). Weiter führen sie aus:

> „Diese Haltung bewahrt eine Art lächelnde Skepsis auch gegenüber den eigenen Träumen und Wünschen, die wohl der beste Schutz gegen aggressive Fundamentalisten und autoritäre Ideologien ist. Sie führt in eine gewissermaßen ästhetische Existenz, in der, mangels anderer Autoritäten, die Sorge um die eigene Autonomie und die private Vervollkommnung im Mittelpunkt stehen". (Breuer et al. 1996)

In diesem Sinne sind die folgenden Überlegungen als Beitrag zur Selbstaufklärung der Open-Source-Bewegung zu verstehen; implizit wird damit behauptet, dass es auch hier um die klassischen Themen der „Mündigkeit/Unmündigkeit, Autonomie und Kreativität" geht.

Als erste Facette in einer Reihe von weiteren geplanten Beiträgen wird hier Open Source als geistige und soziale Bewegung in der Tradition der Wissens- und Wissenschaftssoziologie in den Blick genommen. Von der Spiegelung der gegenwärtigen Situation an klassischen Fragestellungen in der Vergangenheit versprechen wir uns nicht nur eine differenzierende Wahrnehmung und Sprache, sondern angesichts der aufgeregten Diskussion auch ein wenig mehr Gelassenheit (Vgl. Barbrook 1998).

2. Das Phänomen Open Source oder „Was ist Open Source?"

Wer heute mit einer gewissen Distanz – gleichsam von außen wie eine Ethnologin – die Welt der Informatiksysteme und ihre leibhaftigen Akteure betrachtet, dem erscheint die Welt der informatischen Systeme gleichsam durch zwei recht unterschiedliche Ethnien besiedelt:

Zum einen finden sie Menschen vor, die an dem zweiten Medienumbruch teilnehmen, indem sie Hardware in der Regel zusammen mit Software kaufen und damit angesichts der behaupteten „permanenten Innovation" in eine Spirale des Konsums einsteigen. Da Informatiksysteme universal nutzbar sind, wenn geeignete Software zur Verfügung steht und Software ohne großen Aufwand kopiert und transportiert werden kann, ist die Nutzung *nicht gekaufter* Software an der Tagesordnung.

Auf der anderen Seite finden die Ethnologen Menschen – „Opensourcianer" –, die sich auf der Basis frei zugänglicher Quellcodes der kollaborativen Entwicklung von Software verschrieben haben – in den selteneren Fällen persönlich bekannt, in der Regel eher als *invisible colleagues*. Sie erscheinen als solche in ihren Ritualen Verwandte des Stammes der Wissenschaftlerinnen zu sein.

Wenn diese metaphorische Beschreibung in die richtige Richtung zielt – und davon gehen wir aus – ist es nahe liegend, sich für die weitere Analyse und Beschreibung der diesbezüglichen Aussagen der Wissens- und Wissenschaftssoziologie zu vergewissern.

Dabei ist wie in vielen anderen Bereichen auch hier durch den Zweiten Weltkrieg ein markanter Bruch zu verzeichnen. Während wissenssoziologische Fragestellungen zunächst auch im deutschen Raum zu Hause waren – man denke nur an Max Scheler oder Karl Mannheim, die ihrerseits wieder auf lange Traditionen zurückgreifen konnten – war die weitere Entwicklung eher durch amerikanische Wissenschaftler bestimmt, was nach dem Kriege zu einem Reimport alter Denkansätze führte.[2]

2.1. Open Source als Soziales Subsystem

Seit Luhmann kann heute nicht einfach und allgemein verständlich über Systeme geredet werden. Um dennoch verstanden zu werden, geben wir ein Definitionsangebot aus der Zeit vor Luhmann an. Norman W. Storer hat *soziales System* als eine stabile Folge von Interaktionsmustern definiert, „die sich um den Austausch eines qualitativ einzigartigen Gutes organisiert haben und von einem Satz gemeinsamer Normen geleitet werden, die die fortwährende Zirkulation dieses Gutes erleichtern". Da es in einer Gesellschaft nur eine begrenzte Anzahl qualitativ unterschiedlicher Güter von verbreitetem und andauerndem Interesse gibt, lässt sich mit Talcott Parson im Hintergrund ein überschaubarer Kreis von Subsystemen herausgliedern: das ökonomische, das politische, das religiöse und das Familiensystem. Vereinfachend und sicher erläuterungsbedürftig organisieren sich diese Subsysteme um die Güter Geld, Macht, Sinn und Liebe, ausgehend von einem verbreiteten und dauerhaften Drang zur Kreativität, der seine Befriedigung durch Austausch mit anderen findet. „Mit anderen Worten: Es gibt ein weit verbreitetes und anhaltendes Interesse an dem Gut *Reaktion auf Kreativität*, so dass es die Grundlage eines fünften sozialen Systems[3] innerhalb der Gesellschaft bilden kann" (Storer 1972). Über diese Bemerkung hinaus ist nicht nur an die Wissenschaft, sondern auch an die Künste oder auch an Open Source zu denken.

Dabei ist exkursartig ein möglicher Unterschied in der Art der Reaktion auf Kreativität, in der Art der Anerkennung der Leistungen von Wissenschaftlerinnen und Open-Source-Entwicklerinnen, zu reflektieren.

Anerkennung in wissenschaftlichen Kontexten schlägt sich nieder in der *(dokumentierten)* Wahrnehmung der eigenen Position. Dies hat zur Konsequenz, dass Wissenschaftlerinnen ein großes Interesse daran haben, dass ihre Veröffentlichungen zitiert werden. Dies kann zu Verzerrungen führen. Im Wissenschaftskontext wird allenthalben *zirkulär* zitiert, um das eigene Ranking zu erhöhen – es gibt *Ingroups*, die sich gegenseitig *unterstützen*, da sie beispielsweise in einem sehr spezialisierten Segment forschen (jeder kennt jeden). Die in den Zitationsindices häufig auftauchenden Personen sind nicht unbedingt fachlich herausragend. Neben konkreten personalen Abhängigkeiten kennt die Forschung den sogenannten Matthäus-Effekt: „Wer hat, dem wird gegeben".

In der Informatik findet die erfolgreiche Entwicklung von Informatiksystemen keinen direkten Eingang in Zitationsrankings. Weitere kritische Elemente lassen sich

2 Siehe hierfür Weingart (1972), aber auch Meja und Stehr (1982).

3 Die Autoren würden formulieren: „Grundlage von weiteren Systemen".

in den einschlägigen Darstellungen zur Problematik von Zitationsquantifizierungsversuchen in großer Zahl ausmachen. Exemplarisch seien hier Mattern (2002), Stock (2000) und Weingart et al. (1998) genannt. Zusammenfassend kann festgestellt werden, dass die quantitative Bestimmung der Leistung(-sfähigkeit) von Forscherinnen sehr kritisch betrachtet werden muss. Dennoch wird mit Hilfe der zunehmend öffentlich verfügbaren Rankings die vermeintliche Leistungsfähigkeit einer Forscherin dokumentiert.

Im Open-Source-Umfeld gibt es kein Ranking in dieser Form, als konstruktive Leistung wird primär „die Entwicklung von Code" anerkannt: funktionierender, wartbarer, verständlicher Quellcode, der möglichst auch gut dokumentiert ist und (im besten Fall – trotz der oben angeführten Richtungsentscheidung bei GNU/Linux) aktuellen Erkenntnissen der Fachwissenschaft genügt. Darüber hinaus werden Open-Source–Projekte mit der Zeit durchaus von mehreren Personen geleitet. Über die Lizenz werden die Beiträge aller an dem Projekt konstruktiv mitarbeitenden – inklusive der Entwicklungsgeschichte – dokumentiert. Diese Form der Anerkennung, bis hin zu der Erwähnung, dass Personen zur Identifizierung und Behebung von Fehlern beigetragen haben, ist Antrieb und Herausforderung zugleich.

Um diese Erkenntnis auf den Punkt zu bringen: Was funktioniert, lässt sich durchsetzen/setzt sich durch/wird durchgesetzt.

Das mit Open Source konkurrierende soziale Subsystem des Marktes von Angebot und Nachfrage steht hier nicht im Vordergrund unseres Interesses, es dient eher als Hintergrundfolie, um die Eigentümlichkeiten der Open-Source-Bewegung umso schärfer zu konturieren. Auch die Nutzerinnen von Informatiksystemen nutzen diese Systeme letztlich, um einer gesellschaftlich geforderten Zielvorstellung zu entsprechen. Damit hoffen sie, als Gewinner in dem zweiten Medienumbruch dazustehen, gleichsam als *global player* in der Welt der informatischen Systeme. Angesichts solch hoher Zielvorgaben und faktisch knapper finanzieller Ressourcen beginnen Menschen zu *organisieren*, es kommt zu der von Merton beschriebenen Anomie in Subsystemen der Gesellschaft, wenn nicht in der Gesellschaft überhaupt (Vgl. Merton 1972).

Offensichtlich werden die Akteure der Open-Source-Bewegung von anderen Normen begleitet, von Normen, die in ähnlicher Weise auch das System der Wissenschaft steuern.[4] Nachdem lange in idealistischer Denktradition der Zusammenhang des Wissens mit der Sozialstruktur geleugnet wurde, hat Merton erstmals einen, die *scientific community* konstituierenden Normenkatalog mit dem Anspruch eines Ethos der Wissenschaft aufgestellt, der in der Folgezeit ergänzt, variiert, aber in der Substanz unangefochten blieb. Die von Merton aufgestellten Normen sind:

- der Universalismus
- der organisierte Skeptizismus
- der Kommunismus
- die Uneigennützigkeit

4 Auf die nötige Abgrenzung von Begriffen wie Norm, Werte o. ä. muss hier verzichtet werden. Wir wählen den Begriff Norm, um einen stärkeren Handlungsbezug hervorzuheben.

Diese Begriffe werden wegen ihrer möglichen Missverständlichkeit erläutert und auf die Open-Source-Bewegung bezogen.

Unter *Universalismus* versteht Merton als Kind seiner Zeit, die durch den Ost-West–Konflikt geprägt war, die Annahme, „dass Wahrheit und Wert einer wissenschaftlichen Aussage von den Charakteristika ihres Autors unabhängig sind" (Storer 1972, S. 62). So dürfte diese Aussage auch erkenntnistheoretisch heute nicht mehr haltbar sein, man betrachte nur die Nachbeben des Konstruktivismus. Andererseits ergibt sich für Open Source hier eine interessante Fragestellung: Zeigt die Open-Source-Bewegung eine Tendenz zur Universalisierung? Lassen sich in einer dialektischen Gegenbewegung kulturspezifische Regionalisierungstendenzen feststellen? – ein nach Hermann Lübbe immer wieder zu beobachtender Motor der Geschichte. Wir denken hier beispielsweise an die unterschiedlich sich entwickelnden GNU/Linux-Distributionen von SuSE (vormals deutsche Distribution), Red Hat Fedora (amerikanische Distribution), Mandrake (französische Distribution), Red Flag Linux (chinesische Distribution).

Unter *organisiertem Skeptizismus* versteht Merton, „dass jeder Wissenschaftler selbst verantwortlich gemacht wird zu prüfen, dass frühere Forschungen anderer, auf denen seine Arbeit aufbaut, richtig sind", er sieht damit auch die Pflicht verbunden, Kritik an der Arbeit anderer zu veröffentlichen, wenn der Wissenschaftler diese Arbeit für falsch hält (Vgl. Storer 1972, S. 63). Auch hier gilt es mit Blick auf Open Source, differenzierende Fragen zu stellen: Wenn die verschiedenen Subsysteme nicht nur über verschiedene Güter, sondern auch durch eine unterschiedliche Semantik gekennzeichnet sind, stellt sich die Frage nach der der Open-Source-Bewegung eigenen Semantik und zwar sowohl hinsichtlich der kognitiven als auch der sozialen Dimension. So ist das Wissenschaftssystem üblicherweise durch die Semantik „richtig/falsch" gekennzeichnet. Diese Attribuierung ist unter Berücksichtigung der genannten Relativierung von Wahrheitsansprüchen bei der Affinität zur Mathematik und den Naturwissenschaften auch auf die Open-Source-Bewegung zu übertragen. Das in der Open-Source-Bewegung produzierte Wissen hat darüber hinaus andere binär zu beurteilende Qualitäten: funktional/nicht funktional oder sogar im ästhetischen Sinne: schöne und hässliche Qualitäten. Beispiele für solche Beurteilungen sind:

– Zu der Kategorie „richtig/falsch" kann als Illustration die Auseinandersetzung zwischen Linus Torvalds und Andrew Tanenbaum betrachtet werden: Der Entwurf von GNU/Linux entspricht zum Zeitpunkt der ersten Planung nicht etwa dem aktuellen Stand der Betriebssystemforschung, wie Tanenbaum in der Diskussion deutlich zum Ausdruck bringt. Er kann Linus allerdings nicht davon abhalten, das Betriebssystem in dieser Weise zu konzipieren (Vgl. Tanenbaum und Torvalds 1992). Inzwischen zeichnen sich die Grenzen monolithischer Betriebssysteme zunehmend deutlich ab und führen zu Strategien, das Betriebssystem stärker zu modularisieren.[5]

5 Es stellt sich allerdings die Frage, ob Linux als Betriebssystem existieren würde, wenn seinerzeit Linus nicht den *monolithischen Weg* gegangen wäre, denn: Es sollte ein Betriebssystem entwickelt werden, kein Forschungsprototyp.

– In Mailinglisten zu Softwareprojekten gibt es häufig „Streit um den richtigen Weg". Die Auseinandersetzung wird beispielsweise dadurch ausgetragen, dass Quellcode ge*forkt* wird, d. h. aus bestehendem Quellcode wird eine neue Version ausgekoppelt. Eine erfolgreiche Variante hat die Chance, später wieder in den Hauptzweig aufgenommen zu werden. Die Entwicklung des ersten *Journaling File Systems* für Linux durch Hans Reiser hat die Entwicklung des *Extended File Systems 3*[6] kräftig befördert.[7]

– Gerade im Zusammenhang mit der Einführung neuer Funktionalität, die an gewisse Hardware gebunden ist, stellt sich immer wieder die Situation, dass mit der Methode *quick and dirty* Funktionalität verfügbar gemacht wird. Die so dazu gewonnene Funktionalität wird nach und nach in einem offenen Code-Review und der zunehmend sich verbreiternden Masse an Nutzerinnen sehr kritisch einem Test unterzogen, da es regelmäßig Nutzerinnen gibt, die gewisse Funktionen unbedingt nutzen wollen, auch wenn dies nur um den Preis der Aufgabe der unter GNU/Linux bekannten Stabilität möglich ist: Sie wollen nun mal beispielsweise ihren günstig erworbenen USB-Stick auch unter GNU/Linux nutzen können.

Unter *Kommunismus* oder *Kommunalität* versteht Merton, dass Forschungsergebnisse – in dem hier betrachteten Kontext also Softwareentwicklungen – frei und ohne Begünstigung anderen Wissenschaftlern – hier also Entwicklerinnen, aber auch Benutzerinnen als potenziellen Entwicklerinnen – mitgeteilt werden: „ [. . .] denn Wissen, das nicht an die Öffentlichkeit gelangt, kann nicht Teil anerkannten Wissens sein, an dem die Kreativität gemessen wird und auf das andere Wissenschaftler (Entwickler/-Nutzer) sich in ihrer Arbeit beziehen" (Merton lt. Storer 1972, S. 64). Freie Mitteilung der Arbeitsergebnisse wird hier als eine entscheidende Voraussetzung für den Austausch von kreativen Programmen und Anerkennung dieser Kreativität angesehen. Auch hier wird mit Blick auf die Differenzierung von Quellcode und Weiterentwicklungen nachzufragen sein: Es gibt auch in anderen Bereichen eine kommerziell umarmte Kreativität. Wer möchte den Hobbymalerinnen, die ihre Farben nicht selber herrichten, sondern im Baumarkt kaufen, die Kreativität absprechen?

Im Unterschied zu Hobbymalern sind alle Entwicklerinnen von Open-Source-Bausteinen abhängig von den (funktionierenden) Ergebnissen vorgängiger Entwicklerinnen und damit gezwungen, in einem Netzwerk konstruktive Leistungen kollaborativ zu entwickeln. Selbst wenn diese Aktivitäten scheinbar isoliert stattfinden, so gilt der Wahlspruch „standing on the shoulders of giants". Sehen wir uns die quantitative Komplexität moderner Informatiksysteme näher an. Es wird quasi parallel sowohl

6 *ext3* gehört zu der Kategorie der Journaling File Systeme.

7 Unter http://www.kerneltraffic.org/kernel-traffic/kt20000710_75.html#1 findet sich die Auseinandersetzung über die Integration von ReiserFS in den Linuxkern. Unter http://www.kerneltraffic.org/kernel-traffic/kt20000103_49.html#3 verdeutlicht die Diskussion um die Entscheidungsfindung, welches Dateisystem Eingang in den Linuxkern finden soll. Die Diskussion um die konstruktive Weiterentwicklung ist nicht abgeschlossen, wie unter http://www.kerneltraffic.org/kernel-traffic/kt20030910_231.html#13 nachgelesen werden kann:
„At this point other folks came into the discussion, and the thread ended inconclusively."

am Fundament, aber auch an vielen Dachgauben gearbeitet, eine neue Straße gebaut, nebenbei arbeiten einige auch noch unterhalb des Hauses, weil dort Gold oder Öl vermutet wird. Solcherart Arbeiten führen zu maximaler Kooperationsnotwendigkeit – verteilte Teamarbeit, Code-Review, Kohorten von Personen, die bereit sind, die Stabilität ihrer Systeme aufzugeben, um neue Funktionalitäten auf ihre Verträglichkeit und Stabilität zu testen, Gruppen von Personen, die die Dokumentation weiterentwickeln. Und all dies verteilt über den gesamten Planeten Erde. Ein interessanter Aspekt bei dieser Art der Arbeit ist die Art der Verantwortlichkeit, die jeder Beteiligten für *das Ganze* zukommt. Da niemand diese Komplexität „beherrschen" kann, wird permanent konstruktiv „gestritten", es werden Entscheidungen getroffen, um die sprichwörtliche Stabilität zu erzielen, die Linux gegenüber proprietären Systemen auszeichnet.

Unter *Uneigennützigkeit* versteht Merton (nach Storer 1972, S. 64) eine Norm, die die „Wissenschaft um ihrer Selbst willen bestärken, Wissenschaft und Forschung zum Selbstzweck machen will." Dies mag idealistisch klingen und steht in einem scheinbaren Widerspruch zu dem Tausch der bereitgestellten kreativen Leistung gegen Anerkennung.[8]

Es dürfte deutlich geworden sein, dass diese Überlegungen eher differenzierende Fragen aufwerfen als Antworten geben wollen. Überleitend zur Frage der gesamtgesellschaftlichen Bedeutung der Open-Source-Bewegung sei an die wechselnde Wertschätzung erinnert, die die Beschreibung einer Gesellschaft als Tauschgesellschaft erfahren hat.

Unter dem Titel „Gift-Giving as an Organizing Principle in Science?" (Hagstrom 1965), zitiert in Barnes (1972) hat Warren O. Hagstrom diese Tauschmechanismen vor dem Hintergrund einer Befragung von Wissenschaftlern genauer beschrieben.

> „Manuscripts submitted to scientific periodicals are often called *contributions* and they are, in fact, gifts. Authors do not usually receive royalties or other payments, and their institutions may even be required to aid in the financial support of the periodical. On the other hand, manuscripts for which the scientific authors do receive financial payments, such as textbooks and popularizations, are, if not despised, certainly held in much lower esteem than articles containing original research results [. . .] "

Diese Aussage hat nach unserem Eindruck auch heute noch in weiten Bereichen der Wissenschaft ihre Gültigkeit. Hagstrom legt in Anlehnung an Smelser nahe, dass es beim Tausch immer auch um mehr geht:

> „ [. . .] that the gift mode of exchange is typical not only of science but of *all institutions concerned with the maintenance and transmission of common values, such as the family, religion, and communities.*[9] In general, the acceptance of a gift by an individual or a community implies a recognition of the status of the donor and the existence of certain

8 In der Tat wäre eine hier nicht zu leistende Rückbesinnung auf idealistische Reflexionen über das Reich der Notwendigkeit, der Schönheit und Freiheit etwa von Friedrich Schiller möglicherweise hilfreich.

9 Hervorhebung durch die Autoren

> kinds of reciprocal rights. These reciprocal rights may be to a return gift of the same kind and value, as in many primitive economic systems, or to certain appropriate sentiments of gratitude and deference. In science, the acceptance by scientific journals of contributed manuscripts establishes the donor's status as a scientist indeed, status as a scientist can be achieved only by such gift-giving—and it assures him of prestige within the scientific community."

Damit stehen wir vor der Frage, für welche Werte[10] die Open-Source-Bewegung steht. Welche wechselseitigen Verpflichtungen werden von ihren Akteuren erwartet?

Unter dem Stichwort „intrinsically gratifying" thematisiert Hagstrom neben der soziologischen auch eine psychologische Dimension dieses Austausches. Er ist offensichtlich fest davon überzeugt: „Most scientists have sincere interests in the advancement of science, more than in their own recognition." Diese idealistisch anmutende These dürfte je nach anthropologischen Vorannahmen kontrovers gesehen werden. Gleichwohl wäre es interessant, die Akteure der Open-Source-Bewegung einmal zu diesem Aspekt zu befragen.

Hier soll das Augenmerk auf den Vergleich mit *many primitive economic systems* gelenkt werden. Die Semantik von *primitive* ist ambivalent besetzt, negativ im Sinne von roh, unkultiviert, andererseits aber durchaus positiv im Sinne von grundlegend oder in der Kunst im Sinne von frühe Meister. Diese semantische Ambivalenz verweist auch auf unterschiedliche philosophische Sichtmöglichkeiten. Während im Kontext meist eurozentrierter Fortschrittsphilosophien und -ideologien das Frühere oft auch das Primitivere und Unkultiviertere war, haben wir unter dem Einfluss des Strukturalismus gelernt, Naturvölker weniger als Vorform, sondern als äquivalente Lebensentwürfe zu lesen und den Genozid und „Kulturozid" der letzten fünf Jahrhunderte als Verlust zu begreifen.[11]

2.2. Exkurs über die Konkurrenz

Die Open-Source-Bewegung wird in der Öffentlichkeit vor allem und zuerst als David im Konkurrenzkampf, mit dem auf Klientenseite marktbeherrschenden Giganten,[12] in den Blick genommen.

Aufgeregtheiten und Grabenkämpfe deuten auf Unverständnis und Neues hin. Die Bewahrenden verteidigen ein erfolgreich scheinendes und funktionierendes Paradigma gegen eine Weltsicht, die zuerst einmal bekannte Sicherheiten und Geschäftsmodelle

10 *maintenance and transmission of common values*

11 Wenn wir das schwierige Problem einer quantitativen Erfassung einmal beiseite lassen, so gab es zur Zeit der conqista 4 000 Kulturen auf der Welt, „die sich genügend voneinander unterschieden, um als eigenständig angesehen zu werden: [. . .] Heute, 500 Jahre später, hat sich die Zahl der Kulturen weltweit auf 500 reduziert [. . .] . Das bedeutet einen Verlust von 88% der kulturellen Vielfalt der Welt [. . .] " (Vgl. Galtung 1993, S. 11).

12 Vor der „Machtergreifung" auf PCs durch das Betriebssystem MS-DOS (und folgende, bis hin zu den kaum noch zählbaren Versionen von Windows) hieß dieser Gigant IBM, der sich zur Zeit – sicher nicht uneigennützig – nach dem ökonomischen Scheitern des technisch durchaus ausgereiften OS/2 für Linux engagiert.

in Frage stellt. Damit wird tatsächlich existierenden, profitablen *Seilschaften* mit einem noch nicht etablierten Weg (dem neuen Paradigma, das immer erst nachträglich als ein solches identifiziert werden kann) die Existenzberechtigung abgesprochen. Dies führt auf beiden Seiten tendenziell zu nicht rational erklärbaren Verhaltensweisen.

> „Es ist bemerkenswert, dass sich mit der Open-Source-Bewegung gerade in den Vereinigten Staaten als dem Inbegriff der kapitalistischen Welt ein Ansatz etabliert, der deutlich kommunistische Züge trägt. So stellen die Mitglieder der Open-Source-Bewegung das Ergebnis ihrer unentgeltlichen Arbeit der Allgemeinheit kostenlos zur Verfügung. Aus dieser Sicht verwundert es wenig, dass gerade Microsoft und dessen Gründer Bill Gates als Prototyp des Kapitalisten als Hauptfeindbilder fungieren. Folge dieser politischen Sicht der Open-Source-Bewegung sind zwei Hauptströmungen in der Bewegung. Zum einen gibt es die Ideologen, für die Open-Source-Software fast schon einer Weltanschauung gleicht. Auf der anderen Seite gibt es aber auch die Pragmatiker, denen wohl vor allem die breite Diffusion der Open-Source-Software in den letzten Jahren zuzuschreiben ist." (Acker und Hettich 2001, S. 2)

Wird diese sichtbar gewordene Konkurrenz nicht nur als eine Sichtweise eines ökonomischen Sachverhalts abgetan, sondern philosophisch-gesellschaftswissenschaftlich vertieft, stellt sich die Frage nach der Bedeutung der Konkurrenz für einen gesellschaftlichen Produktionsprozess, der immer auch ein geistiger ist. „Die Bedeutung der Konkurrenz im Gebiete des Geistigen" lautet der Titel eines Vortrages, den Karl Mannheim 1928 auf dem Sechsten Deutschen Soziologentag in Zürich hielt (Vgl. Mannheim 1928). Dieser Vortrag kann in dem oben angegebenen Sinne von Luhmann als klassisch angesehen werden. Heute würden wir wohl eher von der „Bedeutung der Konkurrenz auf dem Gebiet des Wissens" sprechen. Der Rückgriff auf Mannheims Kategorisierung verspricht eine zunächst verfremdende Sichtweise und eröffnet weitere Fragemöglichkeiten auf dem Wege zur Selbstaufklärung der Open–Source-Bewegung.

Nach Mannheim ist die Konkurrenz nicht ein Phänomen der ökonomischen Sphäre, sondern genuin ein Phänomen des gesamten gesellschaftlichen Lebens. Der Begriff Konkurrenz wird nicht aus der Sphäre des Ökonomischen heraus verallgemeinert, sondern umgekehrt, „als die Physiokraten und Adam Smith die bedeutende Rolle der Konkurrenz im Ökonomischen aufwiesen, entdeckten sie nur eine allgemeine soziale Beziehung im besonderen Elemente des Ökonomischen" (Mannheim 1928, S. 332). Der Konkurrenz kommt also auch für den hier thematisierten gesellschaftlichen Bereich des Wissens eine konstitutive Bedeutung zu – in dem Sinne, dass sie nicht nur „peripher als Antrieb, als Anlass, als Gelegenheitsursache zur geistigen Produktion da ist [. . .], sondern dass ihre jeweilige Form konstitutiv in die Gestalt und in den Gehalt der Kulturobjektivationen und in die konkrete Form der Kulturbewegung hineinragt" (Mannheim 1928, S. 328). Mit seinem wissenssoziologischen Ansatz sieht sich Mannheim in der Nachfolge phänomenologischer und geschichtsphilosophischer Traditionen. Neben der Dynamik und der Morphologie des Wissens

wird bei ihm die konstituierende Bedeutung des historischen Zeitmoments in das Zentrum der Reflexion gerückt. Nicht weiter verfolgt wird hier seine Differenzierung zwischen *seinsverbundenem Denken* und *Bewusstsein* überhaupt. Das Wissen, das sich in der Open-Source-Bewegung artikuliert, wird von uns als informatisches Wissen dem seinsverbundenen Wissen zugeordnet.

Damit eröffnet sich der folgende Fragehorizont:

> Welche Form der Konkurrenz wirkt in welcher Weise konstitutiv in die Gestalt und den Gehalt von Open Source als einer Kulturobjektivation und -bewegung hinein?

Zur weiteren Klärung charakterisiert Mannheim Konkurrenz als Wettbewerb, der unter anderem durch verschiedene Parteien mit gleichen Zielsetzungen gekennzeichnet ist und der tendenziell in einen Konflikt, wenn nicht gar Kampf, aber auch in ein Miteinander münden kann. Diese Charakterisierung verweist indirekt auf die Bezeichnung für ein aktuelles quelloffenes Produkt zur kollaborativen Entwicklung von Software: *subversion* (Collins-Sussman et al. 2004).[13]

Bezüglich der Zielsetzung sei auf die vielfach kritisierte Haltung von Linus Torvalds hingewiesen, die durch den autobiographischen Beitrag „Just For Fun" von Diamond und Torvalds (2002) bewusst nicht ideologisch überhöht, sondern in Form eines Understatements darstellt, dass die Bewegungsmomente keiner Ideologie verpflichtet scheinen. Dass in der *Community* allerdings auch ein anderer Wind weht, zeigt sich in der permanent stattfindenden Auseinandersetzung um die verschiedenen Lizenzmodelle. So wird [La]TEX zur Zeit nicht auf einer der *freien* Entwicklungsplattformen weiterentwickelt, weil die LATEX-Lizenz (LPPL) bisher nicht durch die *Open Source Initiative* zertifiziert wurde. Andererseits ist LATEX eines der ältesten und erfolgreichsten offenen Projekte (Vgl. Schröder 2004).

Mit Blick auf Open Source konnten die verschiedenen Parteien als David und Goliath in einer mythischen Interpretation benannt werden, bis große Unternehmen (allen voran IBM) damit begannen, die *Bewegung* zu unterstützen, um Geschäftsinteressen zu verfolgen. Luthiger verdeutlicht die Dimensionen, die zur Mitarbeit in offenen Projekten führen (vgl. Luthiger 2004). Ein erfolgreiches Softwareprodukt (StarOffice) wurde von der Firma Sun als Open Office unter eine Open-Source-Lizenz gestellt. Darüber hinaus beginnt die Firma Microsoft 2004, erste Codeschnipsel unter einer Open-Source-Lizenz zu veröffentlichen. Hinsichtlich der Zielsetzungen wird hier nun eine These aufgestellt, die die philosophische Dimension des oben angegebenen Fragehorizonts verdeutlicht:

> Den konfligierenden Parteien geht es – von außen betrachtet – darum, ihre jeweilige Weltauslegung zur öffentlich herrschenden Weltauslegung zu machen.

Gegen diese analytische Zuschreibung spricht die Erfahrung mit den vielen verschiedenen Lizenzmodellen der Open-Source-Gemeinde (z. B. GPL, BSD, MPL,

13 Weitere Informationen unter (http://subversion.tigris.org/).

QPL, LPPL).[14] Nicht umsonst gibt es sehr viele, in ihren Details sehr unterschiedliche Open-Source-Lizenzen, häufig ändern Autoren gewisse Kleinigkeiten in den Lizenzbedingungen, die sie subjektiv für wichtig halten. Damit wird deutlich, dass es nicht die eine Linie zwischen proprietär und offen gibt. Auch das macht es „den Proprietären" so schwer, mit offenen Produkten „umzugehen". Wir befinden uns eher in einem Zustand der permanenten Diversifizierung, da jede (freie) Autorin die Lizenz, nach der ihr Werk verfügbar ist, nach individuellen Kriterien formulieren kann. Will eine Autorin allerdings Ressourcen nutzen, so muss sie prüfen, ob ihr Lizenzmodell unterstützt wird.

Mannheim beruft sich auf Heideggers „Man" und führt dessen Ansatz weiter, „dass die öffentliche Auslegung des Seins nicht einfach da ist, sie wird auch nicht ausgedacht, sondern es wird um sie gerungen" (Mannheim 1928, S. 335). Als Beispiel möge die Einflussnahme auf die sogenannten Offenen Standards des Internet durch die großen Konzerne dienen, die im W3-Konsortium inzwischen massiv ihre Interessen durchzusetzen versuchen. Auch die inzwischen in der Liste der Open-Source-Lizenzen auftauchenden Firmennamen stützen diese These.[15] Die entscheidenden Unterschiede in den hier thematisierten Weltauslegungen liegen in den jeweils unterschiedlichen Verständnis zentraler Kategorien wie beispielsweise der des „geistigen Eigentums", Arbeit und Kreativität.

Mag vordergründig von einer Offenlegung des Quellcodes gesprochen werden und scheinbar ein (Streit-)Gespräch über ein Weltsegment geführt werden, mit der Eigentumsfrage ist ein die Welt insgesamt ordnendes Prinzip genannt. Und gerade hier wird deutlich, dass die Autoren sich im Kontext der „Freien" in besonderer Weise ihrem Eigentum verpflichtet fühlen, in dem sie es offen zur Verfügung stellen. Dabei zeigt sich Autorenschaft von einer neuen Seite. Die bekannte Verwertungs- und Abhängigkeitskette wird durchbrochen.

Die Auslegung des Seins kann nach Mannheim (1928, S. 336) idealtypisch zustande kommen:

1. auf Grund eines Konsensus, bedingt durch eine spontane Kooperation der Einzelnen und Gruppen

2. auf Grund der Monopolsituation einer auslegenden Gruppe

3. auf Grund der Konkurrenz vieler Gruppen, die ihre besondere Seinsauslegung durchsetzen wollen

4. auf Grund der Konzentration mehrerer vorher atomisiert auftretender Konkurrenten zu einem Standorte, wodurch sich die Konkurrenz in der Gesamtheit auf wenige immer mehr herrschend werdende Pole reduziert

14 Eine Übersicht findet sich unter http://www.fsf.org/licenses/license-list.html.

15 So existieren von einigen bekannten Firmen eigene Lizenzen, wie z. B. Microsoft Shared Source License, Sun Public License, Nokia Open Source License, Apple Public Source License.

Diese Typen der „Auslegung des Seins" können auch als Typen der *Weltauslegung* von Open-Source-Gemeinschaften gelesen werden. Mannheim gibt hier formale Kategorien vor, die jeweils historisch zu situieren und materiell zu füllen sind.

Der Hinweis auf das Idealtypische impliziert, dass jede konkrete historische Situation als Mischform zu beschreiben ist. Während Mannheim hier in Jahrhunderten denkt, dürfte hinsichtlich Open Source in Jahrzehnten, wenn nicht in Jahren zu denken sein. So scheint gegenwärtig die Phase der Monopolstellung von Microsoft zumindest an den Rändern in Frage gestellt zu werden. Es mag als kühn erscheinen, hier Mannheims Beispiel für eine Monopolsituation heranzuziehen: die mittelalterlich-kirchliche Weltauslegung – grundsätzlich gekennzeichnet durch eine strukturelle Statik. Doch ein Vergleich drängt sich auf: Im Mittelalter waren es die geheiligten Bücher, die nur einer bestimmten Elite zugänglich waren und den *Sensibilitätskreis*, d. h. die Handlungsmöglichkeiten, beschränkten. Heute beschränkt ein nicht offen gelegter Quellcode den Sensibilitätskreis all jener, die eigentlich teilhaben und ihren Lebens- und Arbeitskreis kreativ gestalten und dynamisch weiterentwickeln wollen. Eine ganz besonders brisante Situation ergibt sich, wenn monopolisierte Informationstechnologien in ein allgemeines staatliches Bildungsmonopol implementiert werden. Es dürfte deutlich geworden sein, dass es sich bei der Entwicklung von Open Source also nicht nur um ein ökonomisches Problem, sondern eine für die Konstitution unseres kulturellen Selbstverständnisses zentrale Frage handelt. Die Frage könnte lauten: Erwächst in Open Source die Idee einer anderen Gesellschaft im Sinne der klassischen Utopien von Thomas Morus, Campanella, Bacon über Thoreau und Skinner bis zu den Träumen von 1968?

3. Open Source – die Rückkehr der Utopie?

1998 schreibt Richard Barbrook unter der Kapitelüberschrift „Das verlorene Utopia":

> „Das Netz wird von den enttäuschten Hoffnungen der 60er verfolgt. Weil diese neue Technologie eine weitere Periode schneller Veränderungen symbolisiert, blicken viele zeitgenössische Kommentatoren auf die abgestorbene Revolution von vor 30 Jahren zurück, um Erklärungen dafür zu finden, was jetzt gerade passiert." (Barbrook 1998)

Barbrooks Aufsatz macht trotz seiner polemischen Rhetorik deutlich, dass Open Source sich als soziale Bewegung eigentlich gegen zwei Konkurrenten behaupten muss: zum einen gegen die ökonomische, rechtliche und staatliche Gängelung, zum anderen gegen eine Vereinnahmung durch einen „aristokratischen Anarchismus", den er eng mit den Namen Deleuze und Guattari verbunden sieht. In seiner überzogenen Polemik gegen die „Deleuzoguattarianer" werden interessante Denkanstöße von Deleuze und Guattari vorschnell verschüttet.[16] Diesen wird andernorts nachzugehen sein (Vgl. Balke 1998, S. 116 ff.).

16 Gleichwohl weist Barbrook wohl berechtigt auf die Gefahren eines Elitarismus und Absolutheitsanspruchs hin. „Gefangen in ihren Glaubensregeln können die Schüler von Deleuze und Guattari nicht einmal erfassen, warum das Wachstum des Netzes wirklich ein so subversives Phänomen ist." Das

> „Von Anfang an hat die Geschenksökonomie die technische und soziale Struktur des Netzes bestimmt. Obwohl es vom Pentagon finanziert wurde, konnte das Netz nur erfolgreich weiterentwickelt werden, indem man den Benutzern erlaubte, das System für sich selbst zu erschaffen. Innerhalb der akademischen Gemeinschaft ist die Geschenksökonomie schon lange die Hauptmethode, Arbeit zu sozialisieren. Finanziert von Stiftungen oder dem Staat, veröffentlichen Wissenschaftler ihre Forschungsergebnisse, indem sie Vorträge halten und Artikel beisteuern. Trotz der verstreuten Art dieser Geschenksökonomie im Bildungsbereich erlangen Akademiker ihren intellektuellen Respekt voreinander durch Zitate in Artikeln und anderen Formen öffentlicher Anerkennung. Die Zusammenarbeit vieler verschiedener Wissenschaftler wird nur durch diese freie Verteilung von Information möglich." (Barbrook 1998)

Die Konvergenz zu unserer oben angeführten Argumentation ist deutlich, in Verbeugung vor der Open-Source-Bewegung nennt Barbrook ein weiteres überzeugendes Beispiel für die Kreativitätssteigerung:

> „Die Hi-tech Geschenksökonomie bildet sogar die Spitze der Softwareentwicklung. Zum Beispiel gibt Bill Gates zu, daß Microsofts größter Konkurrent, wenn es um das Zurverfügungstellen von Webservern geht, das Apache Programm ist. [. . .] Weil sein source code nicht durch Copyright geschützt ist, können Apache Server modifiziert, mit Zusätzen ausgestattet und von jedem verbessert werden, der die notwendigen Programmierkenntnisse besitzt. Sharewareprogramme beginnen jetzt, das Herzprodukt des Microsoftimperiums zu gefährden: das Windows Betriebssystem. Ausgehend von dem ursprünglichen Softwareprogramm von Linus Torvalds, baut jetzt eine Gemeinschaft von User-Entwicklern gemeinsam ihr eigenes, nicht gesetzlich geschütztes Betriebssystem: Linux. Zum ersten Mal hat Windows einen echten Konkurrenten." (Barbrook 1998)

Man muss darauf hinweisen, dass einige Formulierungen von Barbrook nicht ganz richtig sind: Apache ist keine Shareware. Der Quellcode ist sehr wohl geschützt, zumindest im Sinne der OS-Lizenz. Jedoch ist anzumerken, dass der Quellcode nicht im ökonomischen Sinne geschützt ist, er also für eigene Zwecke genutzt und verändert werden darf.

Im Gegensatz zu den *aristokratischen Anarchisten* spricht sich Barbrook für eine Koexistenz aus:

> „Gleichzeitig arbeiten Millionen von Menschen spontan miteinander im Netz, ohne die Koordination von Seiten des Staates oder des Marktes zu benötigen. Anstatt ihre Arbeit für Geld einzutauschen, schenken

Phänomenale liegt für Barbrook in der teilweisen Organisation des Netzes nach dem Muster einer Geschenksökonomie – frei von der Korruption der Konsumgesellschaft, aber auch frei von einem Totalitätsanspruch.

> sie ihre Kreationen im Austausch für freien Zugang zu Informationen, die von anderen produziert wurden. Diese Zirkulation von Geschenken koexistiert mit einem Austausch von Waren und Finanzierung durch Steuergelder. Wenn sie online sind, wechseln Menschen andauernd zwischen verschiedenen Formen sozialer Aktivität. Zum Beispiel kann ein User in einer Sitzung zuerst mit einem E-Commerce Katalog einkaufen, dann auf der Website der Gemeinde nach Informationen suchen und dann einem Listserver für Schriftsteller einige Gedanken beitragen. Ohne auch nur bewußt darüber nachdenken zu müssen, wäre die Person nacheinander ein Konsument in einem Markt, ein Staatsbürger und ein Anarcho-Kommunist in einer Geschenksökonomie gewesen. Die *Neue Ökonomie* des Netzes ist eine fortschrittliche Form sozialer Demokratie." (Barbrook 1998)

In der Literatur findet sich für eine solche Mischform bzw. Koexistenz häufig auch die Metapher der Allmende.[17] Aus der Geschichte der Allmende ist jedoch bekannt, dass diese soziale Einrichtung einer sozialen Für- und Vorsorge ein fragiles Gebilde und vielen Begehrlichkeiten ausgesetzt war. Bei aller intuitiven Stimmigkeit des Vergleichs sind hier aber auch differierende Momente zu benennen: Wenn ich mir mit einem anderen eine Weide teile, habe ich nur die Hälfte des Futters zur Verfügung. Wenn ich dagegen mein Wissen mit jemanden teile, habe ich deswegen nicht weniger, vielmehr die Chance auf ein Mehr. Hier wird mit Blick auf Open Source eine Ökologie des Geistes weiterzuentwickeln sein.[18]

Es dürfte deutlich geworden sein, dass die Antwort auf die Frage nach der Rückkehr der Utopie je nach gesellschaftspolitischer Verortung ganz unterschiedlich ausfallen dürfte. Während die von Barbrook attackierten Theorie-Jockeys[19] die Frage bejahen könnten, würden wir hier eine weniger emphatische und verhaltenere Auffassung vertreten. Danach wird über Open Source die Erinnerung (z. B. Allmende) an andere, nicht kapitalistische Lebensformen wachgehalten, gleichwohl wird die Zukunft wohl eher durch eine Koexistenz verschiedener Produktionsformen im oben angegebenen Sinne bestimmt sein. Vielleicht bedeutet dies nach dem Ende des Ost-West-Konfliktes sogar einen Utopieverlust.

17 Vgl. http://de.wikipedia.org/wiki/Allmende, http://de.wikipedia.org/wiki/Tragik_der_Allmende, http://de.wikipedia.org/wiki/Wissensallmende, Reckmann (2004).

18 Siehe hierfür Bateson (1994), Spinner (2000) „Karlsruher Ansatz der integrierten Wissensforschung" (KAW) mit den drei Hauptfeldern des Wissensarten-, Wissensordnungs- und Wissensverhalten-Projekts.

19 Nach Barbrook handelt es sich dabei um einen Amsterdamer Slang für Intellektuelle, die mit Philosophien *cut 'n' mix* betreiben, wie DJs in einem Club.

Literaturverzeichnis

Acker, K. und Hettich, F. (2001), 'Open Source – Non-Profit-Engagement oder Service Geschäft?'. Robert Goecke (Hrsg.) Arbeitsbericht der Veranstaltung „Teledienste – Trendanalyse und Bewertung" am Lehrstuhl für Allgemeine und Industrielle Betriebswirtschaftslehre der Technischen Universität München. http://www.segma.de/vorlesung00/opensource.pdf [1. Nov 2004].

Balke, F. (1998), *Gilles Deleuze*, Campus Einführungen, Campus, Frankfurt.

Barbrook, R. (1998), 'Die heiligen Narren. Deleuze, Guattari und die Hightech Geschenkökonomie', *Telepolis* . Aus dem Englischen von Barbara Pichler – Artikel-Nr. 6344 – http://www.telepolis.de/deutsch/special/med/6344/1.html [31. Okt 2004].

Barnes, B. (Hrsg.) (1972), *Sociology of Science: Selected readings*, Penguin.

Bateson, G. (1994), *Ökologie des Geistes: Anthropologische, psychologische biologische und epistemelogische Perspektiven*, number 571 *in* 'Suhrkamp Taschenbuch Wissenschaft', 5. Aufl., Suhrkamp Verlag, Frankfurt a. M.

Breuer, I., Leusch, P. und Mersch, D. (1996), *Welten im Kopf. Profile der Gegenwartsphilosophie. Band 3: England/USA*, Rotbuch Verlag, Berlin. Kurztext: http://www.momo-berlin.de/Mersch_Welten_3.html [14. Nov 2004].

Collins-Sussman, B., Fitzpatrick, B. W. und Pilat, C. M. (2004), *Version Control with Subversion*, O'Reilly, Sebastopol. http://svnbook.red-bean.com/svnbook/book.pdf [11. Jul 2004].

Diamond, D. und Torvalds, L. (2002), *Linus Torvalds: Just For Fun. Wie ein Freak die Computerwelt revolutionierte.*, Deutscher Taschenbuch Verlag, München.

Galtung, J. (1993), *Eurotopia. Die Zukunft eines Kontinents*, Promedia Verlag, Wien.

Görlich, C. F. und Humbert, L. (2003), Zur Rolle der Informatik im Kontext der mehrphasigen Lehrerbildung, *in* P. Hubwieser (Hrsg.), 'Informatik und Schule – Informatische Fachkonzepte im Unterricht. INFOS 2003 – 10. GI-Fachtagung 17. – 19. September 2003', München, S. 89–99. http://www.ham.nw.schule.de/pub/bscw.cgi/d44685/Informatik_Lehrerbildung_N.pdf [14. Nov 2004].

Hagstrom, W. O. (1965), Gift Giving as an Organisational Principle in Science, *in* 'The Scientific Community', Basic Books, S. 12–22.

Klafki, W. (1991*a*), Grundzüge eines neuen Allgemeinbildungskonzepts. Im Zentrum: Epochaltypische Schlüsselprobleme, *in* 'Neue Studien zur Bildungstheorie und Didaktik: Zeitgemäße Allgemeinbildung und kritisch-konstruktive Didaktik', Beltz Verlag, Weinheim, Basel, S. 43 ff.

Künzli, A. (1986), *Mein und Dein. Zur Ideengeschichte der Eigentumsfeindschaft*, Bund Verlag, Köln. ISBN: 3-7663-0916-1.

Luthiger, B. (2004), Alles aus Spaß? Zur Motivation von Open-Source-Entwicklern, *in* B. Lutterbeck und R. A. Gehring (Hrsg.), 'Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell', Lehmanns Media, Berlin, S. 93–106.

Lutterbeck, B. und Gehring, R. A. (Hrsg.) (2004), *Open Source Jahrbuch 2004. Zwischen Softwareentwicklung und Gesellschaftsmodell*, Lehmanns Media, Berlin. ISBN: 3-936427-78-X, http://www.Think-Ahead.org/ [19. Jun 2004].

Mannheim, K. (1928), Die Bedeutung der Konkurrenz im Gebiete des Geistigen, *in* 'Der Streit um die Wissenssoziologie. Erster Band: Die Entwicklung der deutschen Wissenssoziologie', S. 325 ff.

Mattern, F. (2002), 'Zur Evaluation der Informatik mittels bibliometrischer Analyse', *Informatik Spektrum* **25**(1), S. 22–32. Folien (Dagstuhl, März 2002) http://www.vs.inf.ethz.ch/publ/slides/bibliometroSlides.pdf [6. Nov 2004].

Meja, V. und Stehr, N. (Hrsg.) (1982), *Der Streit um die Wissenssoziologie. Erster Band: Die Entwicklung der deutschen Wissenssoziologie*, Suhrkamp, Frankfurt a. M.

Merton, R. K. (1972), Die Priorität bei wissenschaftlichen Entdeckungen. Ein Kapitel der Wissenschaftssoziologie, *in* P. Weingart (Hrsg.), 'Wissenschaftssoziologie', Vol. 1. Wissenschaftliche Entwicklung als sozialer Prozess, S. 121 ff.

Perlis, A. J. (1982), 'Epigrams on Programming', *SIGPLAN Notices* **17**(9), S. 7–13. http://www-pu.informatik.uni-tuebingen.de/users/klaeren/epigrams.html [7. Nov 2004].

Reckmann, H. (2004), 'Pädagogische und gesellschaftliche Potenziale freier Software am Beispiel von Linux'. http://fsub.schule.de/bildung/potenzial/potenzial.htm [15. Nov 2004].

Schröder, M. (2004), 'Projekt-Hosting für TEX-Entwickler', *Die TEXnische Komödie* **16**(1), S. 26–28.

Spinner, H. F. (2000), 'Karlsruher Ansatz der integrierten Wissensforschung (KAW)'. http://www.rz.uni-karlsruhe.de/~Helmut.Spinner/3/B/IV/index.html [21. Nov 2004].

Stock, W. G. (2000), Was ist eine Publikation? Zum Problem der Einheitenbildung in der Wissenschaftsforschung, *in* K. Fuchs-Kittowski, H. Laitko, H. Parthey und W. Umstätter (Hrsg.), 'Wissenschaftsforschung Jahrbuch 1998', S. 239–282. http://www.wissenschaftsforschung.de/JB98_239-282.pdf [7. Nov 2004].

Storer, N. W. (1972), Das soziale System der Wissenschaft, *in* P. Weingart (Hrsg.), 'Wissenschaftssoziologie', S. 60–81.

Tanenbaum, A. S. und Torvalds, L. B. (1992), Appendix A: The Tanenbaum-Torvalds Debate, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources. Voices from the Open Source Revolution', S. 221–251. http://www.oreilly.com/catalog/opensources/book/appa.html [7. Nov 2004].

Weingart, P. (Hrsg.) (1972), *Wissenschaftssoziologie*, Vol. 1. Wissenschaftliche Entwicklung als sozialer Prozess, Athenäum, Frankfurt a. M.

Weingart, P., Pansegrau, P. und Winterhager, M. (Hrsg.) (1998), *Die Bedeutung von Medien für die Reputation von Wissenschaftlern*, Arbeitsbericht zum Lehrforschungsprojekt, Universität – Fakultät für Soziologie, Bielefeld. http://www.uni-bielefeld.de/iwt/mw/lf/ [7. Nov 2004].

Infrastrukturen der Allmende – Open Source, Innovation und die Zukunft des Internets

BERND LUTTERBECK

(CC-Lizenz siehe Seite 463)

Als Allmende organisierte Gebilde, wie zum Beispiel Straßen können in hohem Maße Innovationen erzeugen. Man kann inzwischen gut begründen, welche Eigenschaften für dieses Potential verantwortlich sind. In der Literatur und vielen politischen Debatten scheint man sich einig zu sein, dass auch das Internet eine Allmende oder ein *Commons* ist. Allerdings zeigt eine Durchsicht der Literatur, dass die Gleichsetzung oberflächlich ist. Weil man die innovationsbegründenden Eigenschaften des Netzes nicht kennt, ist die Rede vom „Internet Commons" noch mehr ein Schlagwort als die gehaltvolle Beschreibung eines Ortes. Der Beitrag schlägt vor, die noch offenen Fragen dadurch zu klären, dass man einen Umweg geht und das Internet als *Infrastruktur* betrachtet. Dadurch wird es möglich, an die langjährigen Debatten und die reichhaltige Literatur um klassische Infrastrukturen anzuknüpfen und eine Analogie herzustellen. Dies wird am Beispiel der Infrastruktur *See* verdeutlicht. Der Beitrag kommt zum Ergebnis, dass die Analogie möglich ist. Allerdings kann die Infrastruktur *Internet* nur verstanden werden, wenn man sie mit ingenieurswissenschaftlichen Konzepten über Kommunikation verbindet. Das Internet kann und muss als Allmende organisiert werden, will man das unbestrittene Innovationspotential des Internets erhalten. Da die Diskussion in Deutschland noch am Anfang steht, werden abschließend einige Maßnahmen vorgeschlagen, mit denen sich kurzfrig Verschlechterungen des Status Quo vermeiden ließen.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Kapitel 6

Open Content

Inhalte wollen frei sein

CLEMENS BRANDT

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Open Source as Culture—Culture as Open Source

Siva Vaidhyanathan

The Open Source model of peer production, sharing, revision, and peer review has distilled and labeled the most successful human creative habits into a techno- political movement. This distillation has had costs and benefits. It has been difficult to court mainstream acceptance for such a tangle of seemingly technical ideas when its chief advocates have been hackers and academics. On the other hand, the brilliant success of overtly labeled Open Source experiments, coupled with the horror stories of attempts to protect the proprietary model of cultural production have served to popularize the ideas championed by the movement. In recent years, we have seen the Open Source model overtly mimicked within domains of culture quite distinct from computer software. Rather than being revolutionary, this movement is quite conservatively recapturing and revalorizing the basic human communicative and cultural processes that have generated many good things.

The "Open Source" way of doing things is all the rage. Companies as powerful and established as IBM boast of using Linux operating systems in servers. Publications as conservative as *The Economist* have pronounced Open Source methods *successful* and have pondered their applicability to areas of research and development as different from software as pharmaceutical research (see Economist 2004, Weber 2004).

It is striking that we have to employ phrases like "Open Source" and "Free Software" at all.[1] They are significant, powerful phrases simply because they represent an insurgent model of commercial activity and information policy. They challenge the entrenched status quo: the proprietary model of cultural and technological production.

But this has only recently been the case. The "Open Source" way is closer to how human creativity has always worked. Open Source used to be the default way

1 Throughout this essay and in all of my work I intentionally conflate these two terms while being fully aware of the political distinction that Richard Stallman emphasizes in his defense of "Free Software". Stallman's point – that "Open Source" invites an emphasis on convenience and utility rather than freedom and community, was important to make in the 1990s. He lost the battle to control the terms, just as he has had to concede the rhetorical convenience and ubiquity of "Linux" instead of the more accurate "GNU/Linux". I am confident that anyone who peers into the history or politics of the Open Source movement will encounter Stallman's persuasive case for freedom and the GNU project's central contribution to the growth of the operating system we now call Linux (see Stallman 1999).

of doing things. The rapid adoption of proprietary information has been so intense and influential since the 1980s that we hardly remember another way or another time. However, throughout most of human history all information technologies and almost all technologies have been "open source". And we have done pretty well as a species with tools and habits unencumbered by high restrictions on sharing, copying, customizing, and improving.

We have become so inured by the proprietary model, so dazzled and intimidated by its cultural and political power, that any common sense challenge to its assumptions and tenets seems radical, idealistic, or dangerous. But in recent years the practical advantages of the "Open Source" model of creativity and commerce have become clear. The resulting clamor about the advantages and threats of Open Source models have revealed serious faults in the chief regulatory system that governs global flows of culture and information: copyright.

The Rise of Proprietarianism

Copyright gets stretched way out of shape to accommodate proprietary software. Copyright was originally designed to protect books, charts, and maps. Later, court rulings and legislatures expanded to include recorded music, film, video, translations, public performance, and finally practically all media that now exist or have yet to be created. Software is special, though. It's not just expression. It's functional. It's not just information. It's action. In some ways, the inclusion of software among the copyrightable forms of creativity has complicated and challenged the intellectual property tradition. Copyright and proprietary software have metastasized synergistically.

The proprietary model of software production arose sometime in the 1970s, when mainframe software vendors like AT&T and Digital started asserting control over their source code, thus limiting what computer scientists could do to customize their tools. This was an insult to and offense against these scientists who were acclimated to the academic and scientific ideologies that privilege openness and non-monetary reward systems. In a much more precise sense we can date the spark of the conflagration between the then-insurgent proprietary model and the then-dominant hacker culture (Open Source, although this term did not yet exist) to Bill Gates' 1976 open letter to the small but growing community of personal computer hackers. Gates warned them that his new company, then spelled *Micro-Soft*, would aggressively assert its intellectual property claims against those who would trade tapes carrying the company's software. Since that date, despite frequently exploiting the gaps and safety valves of copyright protection on its rise to the heights of wealth and power, Microsoft and Gates have worked in correlation if not coordination with the steady valorization of intellectual property rights as the chief locus of worldwide cultural and industrial policy (see Vaidhyanathan 2001, Wayner 2000, Raymond 1999).

According to the proprietary ideology, innovation would not occur without a strong incentive system for the innovator to exploit for commercial gain. *Fencing off* innovations becomes essential for firms and actors to establish markets and bargain

away rights. Because innovation so often concerns the ephemeral, trade regarding innovation concerns excluding other from using, exploiting, or copying data, designs, or algorithms. The Clinton, Bush, and Blair administrations in the United States and the United Kingdom embraced the proprietary model as the key to thriving through the de-industrialization of the developed world, thus locking in the advantages that educated, wired nation-states have over those that have been held in technological and economic bondage for centuries. Proprietary models of innovation policy and market relations can be powerful: witness the remarkable success and wealth of the global pharmaceutical industry, or, for that matter, Microsoft. But these models can be just as powerful with limitations that allow for communal creation, revision, criticism, and adaptability: witness the culture of custom cars or the World Wide Web (see Vaidhyanathan 2004, Lessig 2001, 2004).

In fact, as economist Richard Adkisson argues, the veneration of forceful intellectual property rights as the foundation of innovation and creativity above all other forms has promoted an unhealthy cultural and social condition, once which can generate suboptimal levels of investment, asset allocation, and policy choices. Adkisson indicts the widespread belief that intellectual property rights are the best (perhaps only) of all possible arrangements for innovation by alerting us to the *ceremonial status* these rights have assumed. *"Ceremonial encapsulation occurs when ceremonial values are allowed to alter or otherwise limit the application of technologies instrumental in the process of social problem solving,"* Adkisson writes. Specifically, Adkisson warns that blind faith in high levels of intellectual property protection is of the *future-binding type*, in which technology and mythology act synergistically to legitimize elite control over technologies or other innovative or creative processes (Adkisson 2004).

The Return of the Jedi

Richard Stallman took a stand against the proprietary model long before the rest of us even realized its power and trajectory. A computer scientist working in the 1970s and 1980s for the Artificial Intelligence project at *MIT*, Stallman grew frustrated that computer companies were denying him and other hackers access to their source code. Stallman found he was not allowed to improve the software and devices that he had to work with, even when they did not function very well. More importantly, Stallman grew alarmed that he was becoming contractually bound to be selfish and unkind. The user agreements that accompanied proprietary software forbade him from sharing his tools and techniques with others. As a scientist, he was offended that openness was being criminalized. As a citizen, he was concerned that freedoms of speech and creativity were being constricted. As a problem solver, he set out to establish the *Free Software Foundation* to prove that good tools and technologies could emerge from a community of concerned creators. Leveraging the communicative power of technology newsletters and the postal system, Stallman sold tapes with his free (as in liberated) software on them. By the time enough of his constituency had connected themselves through the Internet, he started coordinating projects and conversations among a diverse and distributed set of programmers (Stallman 1999, Williams 2002).

During the late 1990s a growing team of hackers struggled to build the holy grail of free software: an operating system kernel that would allow an array of programs to work in coordination. The group, let by Linus Torvalds, created a system that became known as Linux. It has since become the chief threat to the ubiquity and dominance of Microsoft (see Torvalds 2003, Raymond 2001).

While Linux and the GNU (Free Software) project have garnered the most attention in accounts of Open Source development, the protocols and programs that enable and empower the e-mail, the World Wide Web, IRC, and just about every other activity on the Internet all emerged from community-based project teams, often ad-hoc and amateur. The resulting protocols are elegant, efficient, effective, and under constant revision. They have empowered both the growth of the proprietary model and the Open Source model of cultural production to reach expansive new markets and audiences (see Bradner 1999, Galloway 2004).

Each of these projects illuminates what Yochai Benkler calls *peer production*. Benkler writes:

> "The emergence of free software as a substantial force in the software development world poses a puzzle for this organization theory. Free software projects do not rely either on markets or on managerial hierarchies to organize production. Programmers do not generally participate in a project because someone who is their boss instructed them, though some do. They do not generally participate in a project because someone offers them a price, though some participants do focus on long-term appropriation through money-oriented activities, like consulting or service contracts. But the critical mass of participation in projects cannot be explained by the direct presence of a command, a price, or even a future monetary return, particularly in the all- important microlevel decisions regarding selection of projects to which participants contribute. In other words, programmers participate in free software projects without following the normal signals generated by market- based, firm-based, or hybrid models." (Benkler 2002)

Economists assumed for decades that firms emerged to lower or eliminate transaction costs and coordination problems. But as it turns out, fast, efficient and dependable communication, guided by protocols both social and digital (a process Benkler calls *integration*), can generate brilliant and powerful tools and expressions. Benkler concludes:

> "The strength of peer production is in matching human capital to information inputs to produce new information goods. Strong intellectual property rights inefficiently shrink the universe of existing information inputs that can be subjected to this process. Instead, owned inputs will be limited to human capital with which the owner of the input has a contractual—usually employment—relationship. Moreover, the entire universe of peer-produced information gains no benefit from

> strong intellectual property rights. Since the core of commons-based peer production entails provisioning without direct appropriation and since indirect appropriation—intrinsic or extrinsic—does not rely on control of the information but on its widest possible availability, intellectual property offers no gain, only loss, to peer production. While it is true that free software currently uses copyright-based licensing to prevent certain kinds of defection from peer production processes, that strategy is needed only as a form of institutional jujitsu to defend from intellectual property. A complete absence of property in the software domain would be at least as congenial to free software development as the condition where property exists, but copyright permits free software projects to use licensing to defend themselves from defection. The same protection from defection might be provided by other means as well, such as creating simple public mechanisms for contributing one's work in a way that makes it unsusceptible to downstream appropriation—a conservancy of sorts. Regulators concerned with fostering innovation may better direct their efforts toward providing the institutional tools that would help thousands of people to collaborate without appropriating their joint product, making the information they produce freely available rather than spending their efforts to increase the scope and sophistication of the mechanisms for private appropriation of this public good as they now do." (Benkler 2002)

Benkler's prescriptions seem like predictions. In recent years the governments of nation-states as diverse as South Africa, Brazil, and the Peoples' Republic of China have adopted policies that would encourage the dissemination of Open Source Software.

More significantly, the Open Source model has moved far beyond software. Musician and composer Gilberto Gil, the culture minister of Brazil, has released several albums under a *Creative Commons* license. Such licenses (under which this paper lies as well) are based on the *GNU General Public License*, which "locks" the content open. It requires all users of the copyrighted material to conform to terms that encourage sharing and building (Dibell 2004).

Other significant extra-software projects based on the Open Source model include *Wikipedia*, a remarkable compilation of fact and analysis written and reviewed by a committed team of peers placed around the world. The scientific spheres have rediscovered their commitment to openness through the movement to establish and maintain Open Access Journals, thus evading the proprietary traps (and expenses) of large commercial journal publishers (Kaiser 2004). By 2004, citizen-based journalism, often known as *Open Source Journalism* grew in importance and established itself as essential element of the global information ecosystem (see Rosen 2004, Gillmor 2004). Such experiments are sure to proliferate in response to the failures (market and otherwise) of proprietary media forms (Kelty 2004).

How Open Source Changes Copyright

Copyright is a limited monopoly, granted by the state, meant to foster creativity by generating a system of presumed incentives. The copyright holder must have enough faith in the system to justify her investment. The copyright holder's rights to exclude are limited by some public values such as education and criticism. This is the standard understanding of copyright law's role and scope. But while acknowledging the interests of the public, it omits the voice of the public itself. In other words, the system cannot thrive if the public considers it to be captured, corrupted, irrelevant, or absurd (Vaidhyanathan 2004).

The rise and success of Open Source models foster a general understanding that copyright is not a single right bestowed upon one brilliant individual author, but is instead a "bundle" of rights that a copyright holder (individual, corporation, organization, or foundation) may license. Most importantly, these experiments and project show that "all rights reserved" need not be the default state of copyright protection. For many, "some rights reserved" serves the interests of creators better than the absolutist proprietary model.

As the rhetoric of Open Source and the politics of traditional knowledge and culture emerge in starker relief within the topography of copyright and cultural policy debates, their themes tend to converge. As anthropologist Vladimir Hafstein describes the tension between copyright systems as dictated by the industrialized world and modes of communal cultural production that are best (albeit not exclusively) demonstrated in developing nations, he uses terms that could just as easily be applied to technological peer production. *"Creativity as a social process is the common denominator of these concepts and approaches,"* Hafstein writes. *"From each of these perspectives, the act of creation is a social act. From the point of view of intertextuality, for example, works of literature are just as much a product of society or of discourse as they are of an individual author or, for that matter, reader."* Traditional cultural knowledge, communally composed and lacking distinct marks of individual authorship, is *"a node in a network of relations: not an isolated original, but a reproduction, a copy,"* Hafstein explains. Nothing about Hafstein's descriptions of the politics of traditional knowledge offers a resolution to that particular source of friction in global intellectual property battles. The converging rhetorics, however, reveal the extent to which innovation and creativity often (perhaps most often) lie outside the assumptions of incentives and protectionism upon which high levels of corporate copyright protection rest (see Hafstein 2004, Himanen 2001).

The Open Source model of peer production, sharing, revision, and peer review has distilled and labeled the most successful human creative habits into a political movement. This distillation has had costs and benefits. It has been difficult to court mainstream acceptance for such a tangle of seemingly technical ideas when its chief advocates have been hackers and academics. Neither class has much power or influence in the modern global economy or among centers of policy decision-making. On the other hand, the brilliant success of overtly labeled Open Source experiments, coupled with the horror stories of attempts to protect the proprietary model have added common sense to the toolbox of these advocates.

Bibliography

Adkisson, R. (2004), 'Ceremonialism, Intellectual Property Rights, and Innovation Activity', *Journal of Economic Issues* **28**(2).

Benkler, Y. (2002), 'Coase's Penguin, or, Linux and the Nature of the Firm', *Yale Law Journal* **112**(3).

Bradner, S. (1999), The Internet Engineering Task Force, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices of the Open Source Revolution', O'Reilly, Sebastapol, CA.

Dibell, J. (2004), 'We Pledge Allegiance to the Penguin', *Wired* **12**(11). Wired http://www.wired.com/wired/archive/12.11/linux.html [28 Dec 2004].

Galloway, A. R. (2004), *Protocol: How Control Exists after Decentralization*, MIT Press, Cambridge, MA.

Gillmor, D. (2004), *We the Media : Grassroots Journalism by the People, for the People*, O'Reilly, Beijing, Sebastopol, CA. 1st ed.

Hafstein, V. (2004), 'The Politics of Origins: Collective Creation Revisited', *Journal of American Folklore* **117**.

Himanen, P. (2001), *The Hacker Ethic, and the Spirit of the Information Age*, Random House, New York, NY. 1st ed.

Kaiser, J. (2004), 'Zerhouni Plans a Nudge toward Open Access', Science Magazine, 3 Sep 2004.

Kelty, C. M. (2004), 'Culture's Open Sources: Software, Copyright, and Cultural Critique', *Anthropological Quarterly* **77**(3).

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

Lessig, L. (2004), *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, The Penguin Press, New York, NY. http://www.free-culture.cc/freeculture.pdf.

Raymond, E. S. (1999), A Brief History of Hackerdom, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices of the Open Source Revolution', O'Reilly, Sebastapol, CA.

Raymond, E. S. (2001), *The Cathedral and the Bazaar : Musings on Linux and Open Source by an Accidental Revolutionary*, O'Reilly, Beijing; Sebastopol, CA. Rev. ed.

Rosen, R. (2004), 'Top Ten Ideas of '04: Open Source Journalism, or „My Readers Know More Than I Do."', New York University, http://journalism.nyu.edu/pubzone/weblogs/pressthink/2004/12/28/tptn04_opsc.html [28 Dec 2004].

Stallman, R. (1999), The GNU Operating System and the Free Software Movement, *in* C. DiBona, S. Ockman und M. Stone (Hrsg.), 'Open Sources: Voices of the Open Source Revolution', O'Reilly, Sebastapol, CA.

The Economist (2004), 'An Open-Source Shot in the Arm?', *The Economist* .

Torvalds, Linus et al. (2003), 'Revolution Os [Hackers, Programmers & Rebels Unite]', [S.l.] Los Angeles, Calif.: Wonderview Productions, Distributed by Seventh Art Releasing. videorecording.

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

Vaidhyanathan, S. (2004), *The Anarchist in the Library : How the Clash between Freedom and Control Is Hacking the Real World and Crashing the System*, Basic Books, New York, NY.

Wayner, P. (2000), *Free for All : How Linux and the Free Software Movement Undercut the High-Tech Titans*, HarperBusiness, New York, NY. 1st ed.

Weber, S. (2004), *The Success of Open Source*, Harvard University Press, Cambridge, MA.

Williams, S. (2002), *Free as in Freedom : Richard Stallman's Crusade for Free Software*, O'Reilly, Beijing; Sebastopol, CA.

Industrial Influences

Tile von Damm, Jens Herrmann und Jan Schallaböck

Seit Jahren zeigt sich die Musikindustrie in der Krise. Dass diese ausschließlich durch File-Sharing und CD-Brennen begründet ist, ist jedoch nicht zutreffend. Dennoch ermöglicht die singuläre Begründung unter dem Deckmantel der Kulturpolitik und der Innovationsförderung eine eilige, industriepolitisch geprägte Gestaltung des Rechtes, der Technik, ihrer Infrastrukturen und der Gesellschaft. Musik-, Film- und Computerindustrie betreiben derzeit, unterstützt durch den Gesetzgeber, mit Hochdruck die Einführung und Verbreitung von Digital Rights Management Systemen. Die Folgen dieser Entwicklung, in ihrer sich nun konkretisierenden Form, sind bislang noch wenig untersucht. Zumal die in Gang gesetzte Entwicklung weit über den Musik- und Filmbereich hinausgeht. Die Entwicklung von Alternativen, die geeignet sind, die vielfältigen Chancen der Digitalisierung konstruktiv umzusetzen, spielt nicht die Rolle, die ihr zukommen sollte. Die Autoren identifizieren aus technischem, politik- und rechtswissenschaftlichem Blickwinkel Schieflagen des derzeitig stattfindenden Prozesses. Nicht nur droht durch das neue Urheberrecht die Kriminalisierung Vieler, auch offenbaren sich ernst zu nehmende Schwierigkeiten für die Entwicklung und Verbreitung von Open–Source-Software.

1. Intro

Die Frage der digitalen Verfügbarkeit von Musik wird derzeit hoch kontrovers diskutiert. Die Vorschläge der Bundesregierung zur Regelung des Urheberrechts in der Informationsgesellschaft (Zweiter Korb) sollen die Rahmenbedingungen für die Online-Verwertung von Musik schaffen. Dass es jedoch in dieser Debatte um weit mehr als nur die Gestaltung des Musikbereiches geht, wird oftmals nicht wahrgenommen.

Die Musikindustrie versucht, weitreichende Auswirkungen auf den Umgang mit dem Internet, den Betriebssystemen und dem PC überhaupt zu erwirken, denn sie schickt sich an, die Tür für die Einführung von Digital Rights Management Systemen (DRM) zu öffnen.

Dies bleibt nicht ohne Auswirkung auf die Verbreitung von freie Software. Denn offene Standards kollidieren mit den derzeit favorisierten proprietären Lösungen.

Die Spezifika von Musik und die Struktur der Industrie begünstigen die Einführung von DRM. Alternativen, wie sie im Bereich freie Software vorliegen, genießen noch nicht die gesellschaftliche Relevanz, um die Einführung von DRM abzuwenden. Dies gilt ebenso für die (wenigen) vorliegenden Alternativen im Bereich freier Musik. Gerade im Bereich der Musik fehlt es an der vollständigen Übertragbarkeit der Freiheiten von Open Source.

2. Beat dis[1]

Die Merkmale der Lizenzen von freie Software lassen sich mit vier „Freiheiten" zusammenfassen:

1. Die Freiheit, ein Programm zu jedem beliebigen Zweck auszuführen;
2. Die Freiheit, die Funktionsweise eines Programms zu untersuchen und es an seine Bedürfnisse anzupassen;
3. Die Freiheit, Kopien weiterzugeben und damit seinem Nachbarn zu helfen;
4. Die Freiheit, ein Programm zu verbessern und die Verbesserung an die Öffentlichkeit weiterzugeben, so dass die gesamte Gesellschaft profitiert (Free Software Foundation 2004).

Seitdem Musik auch in digitaler Form gespeichert und verbreitet werden kann, gibt es Überlegungen, das Konzept von freie Software auch auf Musik anzuwenden.[2] Bislang am erfolgreichsten ist dies der Creative-Commons-Initiative (CC) gelungen.[3] Diese hat ein System von Lizenzen entwickelt, das derzeit an die Rechtssysteme einer Vielzahl von Ländern angepasst wird.[4] Im Gegensatz zur GNU General Public License[5] ist es nicht primär auf Software, sondern auf Texte, Bilder und eben auch auf Musik zugeschnitten. Gleichzeitig versucht die CC-Initiative, wesentliche Grundgedanken freier Softwarelizenzen aufzugreifen.

Die nach den freien Softwarelizenzen erforderliche Offenlegung des Quellcodes ist auf Musik nur schwer übertragbar. Dies gilt insbesondere bei analog produzierter Musik, denn hier gibt es nicht einen standardisierten Quellcode. Zudem lässt sich die menschliche Rezeption, Komposition und Interpretation eines Musikstückes nicht vollständig in Bits und Bytes beschreiben. Bei Open-Source-Software erfüllt die Offenlegung zwei zentrale Funktionen: Zum einen wird die Weiterentwicklung

1 Die Single „Beat Dis" von Bomb The Bass war eine der ersten kommerziell erfolgreichen, ausschließlich gesampelten Songs (Bomb The Bass 1988).

2 Ein früher Versuch war die Aktion „Open Music" vom Linuxtag 2001, die explizit auf freie Software rekurrierte (http://openmusic.linuxtag.org/).

3 Die nicht-kommerzielle Organisation Creative Commons gründete sich 2001, um dem System des „automatically copyrighted" freie und selbstbestimmte Lizenzen entgegenzusetzen (http://creativecommons.org/).

4 http://www.icommons.org/

5 Die GNU General Public License wird bei einer Vielzahl freier Softwareentwicklungen verwendet, unter anderem auch beim freien Betriebssystem Linux.

erleichtert. Zum anderen wird durch Transparenz ermöglicht, die Funktionsweise eines Programms zu untersuchen. Für letzteres ist der Bedarf bei Musik nicht gerade offenkundig. Entsprechend sehen auch die CC-Lizenzen keine Entsprechung zu den freien Softwarelizenzen vor.

Im Gegensatz zu Software ist ein Musikstück ein in sich geschlossenes, originäres Kunstwerk, bei dem fraglich ist, ob es einer Verbesserung in einem objektiven Sinne zugänglich ist. Durch die Bearbeitung eines Musikstückes entsteht etwas Neues, das neben dem Alten steht. Mit Sampling und Remixen sind jedoch neue Kunstformen entstanden, die besonders stark darauf aufbauen, Teile aus bestehenden Werken neu zusammenzufügen. Hierfür bietet Creative Commons eigene Sampling-Lizenzen an, die ausdrücklich die auszugsweise Weitergabe und Verwendung von Musik zu diesem Zweck gestattet.[6]

Grundelement der CC-Lizenzen ist die Nichtkommerzialität (Creative Commons 2004). Zwar darf man ein Musikstück neben einer CC-Lizenz auch kommerziell lizenzieren, die Frage, wie man mit einer Musikkomposition online (neben dem nicht abgedeckten Offline-Bereich) Geld verdienen kann, will die CC-Initiative aber grade nicht beantworten. Ihr Anliegen ist eine „freie Kultur", die auf eine geringe rechtliche und technische Regulierung digitaler Kulturgüter setzt (Lessig 2004). Will ein Rechteinhaber oder Rechteverwerter sich diesem Kulturbegriff nicht anschließen, kommen CC-Lizenzen für ihn nicht in Betracht.

Derzeit wird versucht, einen alternativen Weg zur Vergütung der Urheber zu finden. Neben dem klassischen kommerziellen Vertrieb liegt mit der „Berliner Erklärung zu kollektivverwalteten Online-Rechten" die Idee einer Pauschalabgabe zur Bezahlung des Rechtes auf File-Sharing vor (die sogenannte „Content- oder Kultur-Flatrate"[7]). Eine Online-Verwertungsgesellschaft soll danach Urheber und Verlage entsprechend der gemessenen Nutzung ihrer Werke vergüten.[8] Von staatlicher Seite müsste der rechtliche Rahmen hierfür geschaffen werden. Dieses Modell kollidiert jedoch mit den Vertriebs- und Verwertungsinteressen der großen Tonträgerhersteller, die eine Individualvergütung präferieren (Gebhardt 2004). Es ist daher nicht damit zu rechnen, dass sich dieses Modell durchsetzen wird.

6 Das ursprüngliche Gesamtstück darf nach diesen Lizenzen allerdings nicht weitergegeben werden.

7 (Berlin Declaration 2004)

8 Das Modell der Content-Flatrate ist bislang nur rudimentär ausgearbeitet. Unklar ist insbesondere, ob die rechtliche Umsetzung im Rahmen einer fakultativen oder einer obligatorischen Regelung erfolgen soll. Eine obligatorische (d. h. verpflichtende) Regelung wäre wohl verfassungswidrig und eine fakultative (freiwillige) weitgehend nutzlos. Weiterhin bleibt offen, ob die Urheber finanziell entsprechend der kommerziellen Verwertung entlohnt werden können. Daneben ist die Argumentation der Stärkung der Urheber gegenüber den Verwertern sicherlich nicht von der Hand zu weisen, jedoch ist anzumerken, dass nur ein kleiner Teil der musikalischen Aufnahmen heute tatsächlich Geld akquiriert. Eine Flatrate wird – zumindest im Offline-Bereich – dieses Modell sprengen. Lediglich zwischen 10 % und 15 % aller veröffentlichten Tonträger sind erfolgreich, d. h. spielen entweder Gewinn ein oder sind zumindest kostenrentabel (Wicke 1997, Becker und Ziegler 2000).

3. Pump Up The Volume[9]

Spätestens Anfang 2000, als die Nutzung von Internettauschbörsen und die Verbreitung von CD-Brennern stark anstieg, sah sich die Musikindustrie zum Handeln gezwungen. Die bis heute verfolgte Strategie der Tonträgerindustrie besteht aus zwei wesentlichen Teilen: Zum einen die Entwicklung und Implementierung technischer Schutzsysteme und zum anderen der Ausweitung der Rechtslage auf die „Bedürfnisse der Mediengesellschaft".

Seit vier Jahren ist der weltweite Umsatz innerhalb der Tonträgerbranche rückläufig. Die offiziellen Zahlen des Bundesverbandes der Phonographischen Wirtschaft[10] attestieren einen Rückgang in Deutschland von real 20,9 % (2003) im Vergleich zum Vorjahr (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 8 f.). Die beiden Hauptgründe sind der Branche zufolge das Brennen von CDs sowie die digitale Verbreitung über das Internet, insbesondere über Tauschbörsen.

Als angeblichen Beweis stellt sie die Zahlen der verkauften CD-Rohlinge den verkauften Tonträgern gegenüber. Hierbei führt sie nicht verifizierbare Zahlen für kopierte CDs (350 Millionen) und Downloads aus „illegalen" Quellen (600 Millionen Songs) an (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 7 f.). Dennoch liegt der hochgerechnete Gesamtumsatz der Musikbranche in Deutschland 2003 bei 1,816 Milliarden Euro (Bundesverband der Phonographischen Wirtschaft e. V. 2004, S. 8 f.). Das entspricht noch immer dem Niveau von 1990 und ist sogar doppelt so hoch wie 1985 (Kuri 2003*a*). Kann man also wirklich von einer Krise sprechen?

Der rasanten technischen Entwicklung hat die Musikindustrie keine konstruktive Lösung entgegengesetzt. Die „Napsterisierung" wurde zum Synonym für die beginnende Krise der Tonträgerbranche. Dennoch ist es nicht korrekt, den Umsatzrückgang der Musikindustrie ausschließlich auf die technischen Veränderungen und Möglichkeiten zurückzuführen. Die Gründe für die Krise liegen vielmehr in der Kulmination verschiedener Faktoren, die die Musikindustrie nur am Rande anerkannt hat: Die Musikindustrie ist im hohen Maße von der Rezession abhängig. Als Unterhaltungsindustrie lebt sie vom Freizeit- und Konsumverhalten ihrer Käufer/innen (Harker 1998). Dabei konkurriert der Tonträger mit anderen Medienprodukten und Freizeitaktivitäten. Teilweise substituieren diese den Tonträger bzw. führen zu einem Kaufkraftabfluss (Kulle 1998, S. 197). Die anhaltende wirtschaftlich gespannte Lage in Deutschland ist also ein wesentlicher Faktor für die rückläufigen Verkaufszahlen.

Die Umsätze der Musikindustrie bewegten sich Mitte der 1990er Jahre auf einem bis heute einmalig hohen Niveau. Dies ist vor allem der erfolgreichen Einführung der Compact Disc und der parallelen massiven Ersetzung der Vinylplatte geschuldet. Ein erfolgreiches neues Format konnte seitdem – trotz mehrerer Versuche – nicht

9 Die auf dem kleinen Independentlabel 4AD veröffentlichte Single „Pump Up The Volume" erreichte 1987 die #1 der UK-Charts. Der Song ist einer der maßgeblichen Wegbereiter des Acid House und der Sampling-Culture (M/A/R/R/S 1987).

10 Der Bundesverband der Phonographischen Wirtschaft ist die deutsche Landesgruppe der International Federation of the Phonographic Industry (IFPI). Er repräsentiert in Deutschland 91 % des deutschen Musikmarktes.

eingeführt werden.[11] Zudem konzentriert sich der Markt zu über 70 % auf die fünf größten Plattenfirmen (Becker und Ziegler 2000, S. 20).[12] Ihre starke Stellung wird zusätzlich dadurch begünstigt, dass ein Großteil der Independentlabels über Vertriebs- und Beteiligungsabkommen mittlerweile an die großen Firmen gebunden ist.[13]

Trotz der bis Ende der 1990er steigenden Verkaufszahlen, sah sich die Branche der sinkenden Rentabilität ihrer Produkte ausgesetzt, der sie kein Konzept entgegengesetzt hat. Dies liegt zum einen an der abnehmenden klassischen Aufbauarbeit und langfristigen Förderung von Künstlern zugunsten schneller, Gewinn versprechender Produkte. Beim Abebben eines Trends sind durch die ausbleibende weitere Förderung eines Künstlers die Ausgangsinvestitionen für Produktion und Marketing verloren (Friedrichsen 2004, S. 37). Der daraus resultierende „wirtschaftliche Zwang" führt zu einer steigenden Anzahl der Veröffentlichungen und einem Steigen des Break-Even Punktes.[14] Hinzu kommt, dass – insbesondere für die erfolgreichen Künstler – nicht nur die Kosten für Marketing und Promotion, sondern auch die erfolgsunabhängige Vergütung sowie teilweise die Produktionskosten sprunghaft angestiegen sind.[15]

Die Gründe für die Krise der Tonträgerindustrie sind also vielfältig und hauptsächlich innerhalb der Branche zu suchen. Neben den internen Faktoren haben allerdings die externen Faktoren wesentlich zu einer Beschleunigung der Krise beigetragen. Neben der wirtschaftlichen Rezession sind sicherlich die technischen Möglichkeiten – vor allem das File-Sharing und das CD-Brennen – zu nennen.

4. Blue Monday[16]

Eine wesentliche Stärke der Tonträgerindustrie lag bislang in der Herstellung und dem Vertrieb eines Tonträgers, der aber bei der Distribution von Musik über das

11 Zu nennen ist hier bspw. die von Sony entwickelte Minidisc (MD). Einzig die DVD erreichte in den vergangenen Jahren nennenswerte, stetig steigende Verkäufe, so dass die Phonoindustrie ihr bereits bis 2010 einen Absatz in der Größenordnung der CD prophezeit (Gebhardt 2003).

12 Mit der Fusion von *Sony Music* und der *Bertelsmann Music Group* (BMG) entsteht nach Universal Music die gemessen am Umsatz weltweit zweitgrößte Plattenfirma. Die Anfang August 2004 vollzogene Fusion wird Anfang 2005 umgesetzt (Netzeitung 2004). Neben Sony BMG Music Entertainment bilden somit Universal Music, die Warner Music Group und EMI Music die größten Tonträgerunternehmen, die so genannten Majors.

13 Die Bezeichnung „Independents" oder kurz „Indie" stand ursprünglich tatsächlich für einen von den großen Firmen unabhängigen wirtschaftlichen Status. Seit den 1970er Jahren allerdings ist dies in den meisten Fällen so nicht mehr gültig.

14 Der Break-Even Punkt ist die Grenze, an der eine Produktion Gewinn einspielt. Eine weitere Entwertung der Musik ergibt sich durch die steigende Zahl der formatierten Sendeformen im Rundfunk. Die gesendete Musik reduziert sich auf den kleinsten gemeinsamen Nenner und wird auf seine „Abschaltfestigkeit" hin geprüft. Zunehmend jedoch orientierte sich die Tonträgerindustrie in ihrer Veröffentlichungspolitik am Sendeformat.

15 Ein absurdes Beispiel hierfür ist sicherlich die Auflösung des laufenden Vertrags mit Mariah Carey seitens des Labels Virgin/EMI, nachdem ihr Album „Glitter" mit „nur" vier Millionen verkaufter Exemplare als Flop angesehen wurde. Sie bekam eine Abfindung von 32 Millionen US-Dollar (Lau 2002, S. 51 f.).

16 Oder „How Does It Feel To Treat Me Like You Do". Die Elektronikpioniere New Order veröffentlichten 1983 ausschließlich auf 12" den Klassiker „Blue Monday" (New Order 1983).

Internet nicht zwangsläufig notwendig ist. Vielmehr kann jeder abgekoppelt von einem physischen Tonträger seine Musik über das Internet vertreiben. Das zentrale Geschäftsmodell der Musikindustrie – insbesondere der Majors – ist also bedroht.

Neue kommerzielle Vertriebs- und Verwertungskonzepte, die finanziellen Erfolg versprechen, entstehen nur zögerlich. Zwar haben sich die Majors inzwischen zu globalen Medienunternehmen entwickelt,[17] die neben dem klassischen Tonträgerbereich eben auch digitale Vertriebswege bereithalten. Die Uneinigkeit der Branche, eine fehlende technische und rechtliche Ausgestaltung und die (finanziell) nach wie vor erfolgreiche klassische Verwertung haben bislang keine vernünftigen Alternativvertriebswege ermöglicht. Dieses zeigt auch der lange Weg zu einer gemeinsamen Online-Plattform der Musikindustrie (Kuri 2004). Worüber sich die Tonträgerbranche allerdings einig ist, ist die Forderung, dass beim Musikvertrieb über das Internet die vertriebenen Daten vor unberechtigter Vervielfältigung geschützt werden sollen.

Digital Rights Management (DRM) ist die Technologie, mit deren Hilfe die Durchsetzung von Nutzungslizenzen für digitale Daten ermöglicht werden soll. Mit Hilfe von DRM soll eine der Grundeigenschaften digitaler Daten verhindert werden, nämlich die Möglichkeit, nahezu ohne Aufwand verlustfreie Kopien herzustellen, damit sich auch diese Daten wie Verbrauchsgüter vermarkten lassen.

DRM soll ein „umfassendes Vertriebskonzept für digitale Güter" ermöglichen (Günnewig 2002). Neben der reinen Nutzungs- und Zugangskontrolle (Kopierschutz)[18] sollen DRM-Systeme auch Authentizität und Integrität von Daten sicherstellen, Metadaten zu den übertragenen Informationen liefern und die komplette Verwertungskette vom Bereitstellen der Daten bis zur Übertragung zum Kunden und der Abrechnung verwalten (Fetscherin 2004).

Kritiker nennen DRM auch „Digital Restrictions Management", also Beschränkungsverwaltung (Stallman 2002). Da es um die Verhinderung bestimmter Nutzungen digitaler Daten geht, ist dieser Begriff durchaus zutreffend.[19]

Der bislang erfolgreichste Versuch, Musik über das Internet zu vertreiben, kommt von der Firma *Apple* (Herrmannstorfer 2004). Auf der Apple eigenen Online-Verkaufsplattform erworbene Musik kann man (ohne vorherige Konvertierung) nur auf die von Apple entwickelten und lizenzierten Abspielgeräte kopieren. Auch Audio–CDs lassen sich nur in beschränkter Anzahl herstellen. Dazu wird beim Installieren der Abspielsoftware, die ebenfalls von Apple kommt, auf einem Computer ein für diesen eindeutiger „machine identifier" erzeugt, der diese Beschränkung sicherstellt, wobei das DRM nicht in Abhängigkeit vom Dokument variiert werden kann.

17 Beispielsweise fusionierten 2001 Time Warner und AOL zu dem weltweit größten Medienkonzern.

18 So bei „Lightweight DRM" (Frauenhofer 2003). Hierbei werden die Daten nicht verschlüsselt und technisch kopiergeschützt, sondern lediglich mit einem eindeutigen Wasserzeichen versehen. Anhand dieses Wasserzeichens kann ein Kunde ermittelt werden, der Rechte an einer Datei erworben und die Datei ggf. unrechtmäßig verbreitet hat. Hier entsteht der Schutz vor unrechtmäßiger Vervielfältigung nicht direkt durch technische Maßnahmen. Wie im klassischen Urheberrecht ist der Urheber zur Durchsetzung seiner Interessen auf staatliche Maßnahmen (also Zivilverfahren und Strafverfolgung) angewiesen.

19 Insoweit technisch unterbunden wird, was rechtlich erlaubt ist, ist offenkundig auch die Bezeichnung „Rights Management" irreführend, denn es wird ja gerade nicht das Recht verwaltet.

Microsoft bietet ebenfalls ein DRM-System an, das mit dem Windows eigenen Mediaplayer zusammenarbeitet (Microsoft Deutschland 2004). Dieses System wird auf diversen Download-Plattformen für Musik verwendet (Hansen und Block 2004, S. 176 f.). Im Gegensatz zur Lösung von Apple kann der *Windows Media Rights Manager* beliebige unterschiedliche Lizenzen verwalten.

5. Sour Times[20]

Die beschriebenen Systeme teilen eine gemeinsame Schwachstelle: Die Abspielsoftware auf dem Computer des Nutzers muss in das DRM-System einbezogen sein. Sie entschlüsselt die Daten und gibt sie aus, und sie „entscheidet" bei vorliegenden Schlüsseln auch anhand der Lizenz, ob ein bestimmtes Stück überhaupt bzw. wie oft es abgespielt werden darf.

Dabei besteht die Gefahr, dass die Daten bei der Abspielsoftware in entschlüsselter Form „abgegriffen" werden. Der Schutz, den das DRM-System gegen unrechtmäßiges Verbreiten der Daten bieten soll, ist somit leicht zu umgehen.

Um unter anderem auch diesem Problem Herr zu werden, gründete sich die *Trusted Computing Group*[21], ein Industriekonsortium, das versucht, ein System zu entwickeln, mit dessen Hilfe es möglich sein soll, auf dem Rechner eines Anwenders nur bestimmte, als sicher bekannte Software zu verwenden. Dabei soll die Hardware des PC um ein Modul (ähnlich einer Smartcard) erweitert werden, mit dem kryptographische Schlüssel gespeichert werden. Diese sind für den Eigentümer des PC unzugänglich, ermöglichen aber einem Kommunikationspartner, also auch dem Online-Anbieter von Musik, sicherzustellen, dass nur im Rahmen der technischen Vorgaben des Anbieters auf die Daten zugegriffen werden kann.

Dabei besteht die Gefahr, dass durch ein proprietäres *Trusted Computing* (TC) die Entwicklung von Open-Source-Software bis zur Unmöglichkeit erschwert werden kann (Stallman 2002). Da jede Anwendung, die in einem sicheren Umfeld ausgeführt werden soll, zertifiziert und signiert werden muss, entstehen für Open Source neue Kosten, die die Entwicklung von Open-Source-Software zumindest für einzelne freie Entwickler drastisch erschwert. Aber auch für reine Windows-Anwender entstehen durch ein proprietäres TC Probleme. Bei einer intransparenten Zertifizierungsstruktur wird dem Nutzer die Selbstbestimmung über seinen eigenen Computer entzogen (Anderson 2004). Es sind offene Standards notwendig, um einen selbstbestimmten Zugang zu digitalen Medien zu gewährleisten.

Gleichzeitig kann eine Technologie wie TC helfen, ein Problem im Zusammenhang von Open Source und DRM zu lösen. DRM dient grundsätzlich dazu sicherzustellen, dass entschlüsselte Daten nur unter den Bedingungen der dazugehörigen Lizenz zugänglich sind. Ein DRM-System basierend auf Open Source muss auch die Routine

20 Mit ihrer 1994 veröffentlichten zweiten Single „Sour Times" und dem Album „Dummy" zählen Portishead zu den Pionieren des TripHop (Portishead 1994).

21 Die *Trusted Computing Group* (TCG) versteht sich selber als „not-for-profit organization". Als „Promoter" fungieren: AMD, HP, IBM, Intel, Microsoft, Sony und Sun (https://www.trustedcomputinggroup.org/).

enthalten, die die verschlüsselten Daten entschlüsselt und weiterverarbeitet. Da der Code offen liegt, kann diese Routine dann dazu benutzt werden, die entschlüsselten Daten aus dem DRM-System zu befreien. Der Schutz ist damit hinfällig. Zwar ist immer noch ein Schlüssel erforderlich, um die Daten zu entschlüsseln, was danach mit ihnen geschieht, liegt allerdings nicht mehr in der Hand derjenigen, die die Verschlüsselung vorgenommen haben.

Anliegen von DRM ist es allerdings, genau dies zu unterbinden. Die klassische Antwort auf dieses Problem ist Closed-Source-Software. Eine andere Lösung wäre es sicherzustellen, dass nur Code ausgeführt wird, der als sicher bekannt (und signiert) ist. Ein solcher Schutz muss dann allerdings durch die Hardware sichergestellt werden. Damit könnten ohne Weiteres die Quellen der Software veröffentlicht werden. Sie können nur nicht modifiziert werden, ohne erneut signiert zu werden. *Trusted Computing* könnte also Open-Source-DRM-Systeme ermöglichen.

6. What time is love?[22]

Aus einem tradierten, überkommenen Verständnis von „geistigem Eigentum", das auf John Locke zurückgeführt wird, hat der Urheber ein „angeborenes Recht" an den von ihm geschaffenen Gütern (Kirchhof 1998, S. 2). Aus dieser vorherrschenden Sicht auf das Urheberrecht ist die Einführung von DRM durchaus konsequent, schließlich ermöglicht es dem Urheber in umfänglicher Weise, darauf Einfluss zu nehmen, was mit seinem Werk geschieht.

Meist wird mittlerweile das Urheberrecht auf die Anspornttheorie gestützt (Oechsler 1998). Das Urheberrecht soll Anreize für geistige Tätigkeit und Schöpfung schaffen. Dies entspricht dem Bild einer auf Fortschritt ausgerichteten Gesellschaft. Der Schutz, den das Urheberrecht dem Schöpfer gewährt, mache es für ihn attraktiver, sein Werk zu veröffentlichen, und gäbe der Gesellschaft die Möglichkeit, von der Leistung des Schöpfers zu profitieren.

Mit dem *Gesetz zur Reform des Urheberrechtes in der Informationsgesellschaft*, dessen „Zweiter Korb" nunmehr als Referentenentwurf vorliegt, geht der Gesetzgeber noch einen Schritt weiter; das Gesetz „bemüht sich . . .um einen Interessensausgleich" (BMJ 2004, S. 19). Dem liegt ein Urheberrechtsverständnis zugrunde, das stark auf Vertragsrecht rekurriert. Wegen der Sozialbindung des (geistigen) Eigentums[23] hat der Gesetzgeber, ähnlich wie etwa im Verbraucherschutz sowie im Arbeits- oder Mietrecht, die Aufgabe, den Schwachen vor dem Starken zu schützen.

DRM bringt eine neue Problemstellung, für die der Gesetzgeber neue Wege finden muss. Vor der Einführung von Tonträgern Anfang des 20. Jahrhunderts war es schon technisch unmöglich, Kopien herzustellen. Ein Recht auf Privatkopie wäre damals

22 „The KLF – What Time Is Love" erschien 1989 auf dem eigenen Label KLF Communications. Das Avantgarde-Projekt „The Justified Ancients of Mumu", alias The KLF (Kopyright Liberation Front) erreichten unter dem Projektnamen The Timelords 1988 mit „Doctorin' The Tardis" die #1 der UK-Charts (KLF 1989, Drummond und Cauty 1988).

23 Art. 14 GG.

also völlig nutzlos gewesen. Technische Neuerungen erweiterten die Handlungsmöglichkeiten der Verbraucher. Mit der Verbreitung von Tonbändern und Audiokassetten wurde es möglich, ohne großen Aufwand Musik privat zu kopieren und weiterzugeben.[24] Mittels DRM werden die neuen Möglichkeiten des Kopierens und Verbreitens von urheberrechtlich geschütztem Material technisch unterbunden. Neu ist also, dass der Verbraucher in seinen Handlungsmöglichkeiten eingeschränkt wird.

Grundsätzlich gilt dabei die allgemeine Handlungsfreiheit[25] auch für den Urheber. Wenn jemand mittels DRM die technischen Möglichkeiten beschränken will, kann er das zunächst tun, ein Verbot von DRM würde einen Grundrechtseingriff darstellen. Im Datenschutz, bei personenbezogenen Daten, wird man dem „Inhaber" das Recht, „seine" Daten zu verschlüsseln, nicht verwehren wollen.[26] Genauso dürfte man sie wohl auch mit DRM ausstatten und so die Entschlüsselung nur für eingeschränkte Zwecke zulassen. Ein Grundrechtseingriff kann aber beispielsweise gestattet sein, wenn man sich in den Geschäftsverkehr begibt und seine Musik verkaufen will. Hier könnte man unter anderem wegen der Sozialbindung des geistigen Eigentums oder der Wissenschaftsfreiheit Eingriffe in die allgemeine Handlungsfreiheit des Urhebers rechtfertigen.

Dieser Weg wird aber mit dem vorliegenden Referentenentwurf nur zögerlich beschritten. Stattdessen stützt er die Geschäftsmodelle, mit denen die Tonträgerindustrie ihre Marktstellung behaupten möchte. So wird versucht, den Down- und Upload in File-Sharing-Börsen unter Strafe zu stellen. Zwar verkündete die Bundesjustizministerin vollmundig, es gehe nicht um die „Kriminalisierung der Schulhöfe" (Stein 2004, S. 4). Doch die angekündigte Bagatellausnahme, die das Downloaden einzelner Songs aus illegalen Tauschbörsen[27] von Strafe ausnimmt, geht an der Realität der Schulhöfe und auch weitaus größerer Bevölkerungskreise vorbei. Denn es werden in aller Regel nicht einzelne Songs aus illegalen Tauschbörsen, sondern hunderte Songs in legalen Netzwerken getauscht (Kuri 2003*b*).

Das Justizministerium nimmt, um der Musikindustrie ihre Vorstellungen von der Online-Verwertung zu sichern, die Kriminalisierung von Millionen von File-Sharing–Nutzern sehenden Auges[28] in Kauf. Dies ist rechtspolitisch wenig wünschenswert. Es besteht eine Diskrepanz zwischen den Nutzungsgewohnheiten Vieler und der

24 Diese Möglichkeit wurde vom Bundesverfassungsgericht legitimiert mit der Begründung, dass das Kopieren nicht zu unterbinden sei ohne weitreichende, nicht legitimierte Eingriffe in die Privatsphäre der Menschen.

25 Art. 2 GG, oder: „Der Mensch ist frei. Er darf tun und lassen, was die Rechte anderer nicht verletzt oder die verfassungsmäßige Ordnung des Gemeinwesens nicht beeinträchtigt" (Land Hessen 1946, Art. 2).

26 Interessanterweise war dies Gegenstand einer langen Debatte. Bei personenbezogenen Daten war lange Zeit die Anordnung einer Schlüsselhinterlegung im Gespräch (Computerwoche 1997).

27 Das Gesetz formuliert die Regelung freilich, ohne direkt auf Peer-to-Peer-Netzwerke Bezug zu nehmen und stellt auch klar, dass der Download auch im Rahmen der Bagatellausnahme widerrechtlich bleibt, so dass zivilrechtliche Ansprüche gegen den Tauschbörsennutzer nicht ausgeschlossen bleiben: „Nicht bestraft wird, wer rechtswidrig Vervielfältigungen nur in geringer Zahl und ausschließlich zum eigenen privaten Gebrauch herstellt" (BMJ 2004, §106, Abs. 1, Satz 2(neu)).

28 Das ist ein Ergebnis einer unveröffentlichten, für das Ministerium entwickelten Online-Befragung zu den Überzeugungen von Tauschbörsennutzern. Sie zeigte, dass etwa ein Drittel der etwa sieben Millionen derzeitigen Nutzer nicht mit Aufklärungskampagnen zu erreichen sind. (Kemmler 2004)

willkürlichen Verfolgung Einzelner.

Bei der Einführung von Tonbandgeräten jedenfalls hat man sich für einen anderen Weg entschieden und das Recht auf Privatkopie eingeführt, um mittels Pauschalabgaben und Verwertungsgesellschaften die Vergütung zu sichern.[29] Heute wird ein ähnliches System gefordert: Die Kultur-Flatrate.

Derzeit ist noch offen, ob es gelingt, mit dem rabiaten Mittel des Strafrechts den Tauschbörsen das Wasser abzugraben. Die Nutzungszahlen deuten nicht darauf hin, ein Rückgang ist bisher nicht feststellbar (Wilkens 2004), obwohl die Musikindustrie beginnt, Klagen gegen die Nutzer anzustrengen und die Filmindustrie mit einer provokanten Werbekampagne[30] versucht, auf die Nutzer einzuwirken.

Neben der Strafbestimmung für das Downloaden sieht das neue Urheberrecht ein Verbot der Umgehung technischer Schutzmaßnahmen vor[31] (BMJ 2004, S. 19). Auch auf diesem Wege begünstigt es die Einführung von DRM-Systemen. Liegt ein Schutzmechanismus vor, werden digitale Beschränkungen des Urheberrechts Makulatur, d. h. der Nutzer darf seine Rechte nicht mehr ausüben, wenn ein technischer Schutzmechanismus vorliegt.[32] Ein Berechtigung zu digitalen Kopien ist nicht durchsetzbar, wenn Kopierschutzmaßnahmen bestehen. Der Gesetzgeber ermutigt die Musikindustrie de facto, die Werke mit Kopierschutz- oder DRM-Systemen zu versehen, da sie sonst die Privilegisierung des neuen Urheberrechts nicht für sich in Anspruch nehmen kann. Zwar werden CDs zunehmend wieder ohne Kopierschutz ausgeliefert, weil die Kunden die mit den kopiergeschützten „Un-CDs“[33] einhergehenden Beschränkungen nicht akzeptieren. Beim Onlinevertrieb zeichnet sich aber ab, dass sich DRM-Verfahren durchsetzen, die die Kopierbarkeit der Daten beschränken.

Der Gesetzgeber stützt also die Einführung von DRM im Rahmen der Reform des Urheberrechtes. Ansätze für eine Regulierung von DRM finden sich aber nicht. Selbst die Diskussion um die Auswirkungen von DRM findet bisher nur sehr zaghaft statt. Lediglich eine Expertenrunde der Europäischen Kommission, die *High Level Group* (HLG) des Aktionsplans „eEurope 2005“, beschreibt die Erfordernisse bei einer breiten Einführung von DRM (High Level Group 2004). Vor allem die erhobene Forderung nach Interoperabilität von DRM auf unterschiedlichen Systemen ist im Zusammenhang mit Open Source von immenser Bedeutung.[34]

---

29 Vgl. BGHZ 42, 118; BVerfG GRUR 1980, 44, 48.

30 Gegenüber dieser hat selbst der Zentralverband der deutschen Werbeindustrie Bedenken geäußert hat (Kuri 2003*c*).

31 Hierbei setzt die Bundesregierung völkerrechtliche Verpflichtungen um, die sie bereits 1996 eingegangen ist (WIPO 1996).

32 Unter die so genannten Urheberrechtsschranken fallen unter anderem das Recht zur Kopie für Zwecke der Wissenschaft und Lehre, das Zitatrecht in beschränktem Umfang, die Privatkopie, die Vervielfältigung für gerichtliche und behördliche Verfahren und andere.

33 Der Begriff „Un-CD“ geht darauf zurück, dass die derzeitig zumeist eingesetzten Kopierschutzverfahren bei CDs mit einer Abweichung vom allgemein vereinbarten CD-Standard ermöglicht werden, was auch dazu führt, dass sie sich auf vielen CD-Playern nicht mehr abspielen lassen.

34 An der HLG wurde bemängelt, dass hier die Interessen der Wirtschaft, insbesondere der Verwertungsindustrie deutlich im Vordergrund standen und die Bedürfnisse der Verbraucher vernachlässigt würden (Grassmuck 2004). So wurde nur eine einzige Verbraucherschutzorganisation, das *Bureau Européen des Unions de Consommateurs* (Europäische Verbraucherschutzorganisation, BEUC), zu den Verhandlun-

Die HLG präferiert eine Lösung auf der Basis offener Standards, damit keine Herstellerabhängigkeiten entstehen. Solche offenen Standards sind in Ansätzen bereits vorhanden, z. B. die *MPEG21 Rights Expression Language*, die eine Beschreibung von erteilten Nutzungslizenzen ermöglicht (Wang 2004). Allerdings sind vor allem Implementierungen erforderlich, um ihre Durchsetzung zu fördern.

Derzeit existieren lediglich „Inseln proprietärer Systeme" (High Level Group 2004), die untereinander inkompatibel sind. Aus diesen kann sich ein De-facto-Standard herausbilden. Dies geschieht in der Regel dann, wenn ein Anbieter sich am Markt durchgesetzt hat. Um Plattformabhängigkeiten zu vermeiden, müsste dieser die Spezifikation des Standards öffnen. Dies ist weder beim Angebot von Apple noch dem von Microsoft der Fall. Eine Portierung auf offene Betriebssysteme scheint unwahrscheinlich. Ein offenes Betriebssystem wie Linux wird also bei der Fortschreibung der derzeitigen Entwicklung keine Alternative zu den proprietären Systemen darstellen können.

7. Outro

Das Geschäftsmodell einer Industrie, die sich selbst noch als „Tonträgerindustrie" bezeichnet, hätte weit weniger Überlebenschancen, wäre diese Branche nicht so stark konzentriert. Diese Konzentration und ihre bereits existierende starke Rechtsposition ermöglicht ihr einen erheblichen Einfluss auf die Politik. Diesen Einfluss setzt sie für die weitere Stärkung ihrer Verwertungsrechte ein und nötigt dabei den Verbrauchern eine DRM-Infrastruktur auf. Die Gefahr besteht, dass es nun zu der Durchsetzung eines proprietären De-facto-Standards kommt, mit weitreichenden Konsequenzen auch auf anderen Gebieten als nur dem Musikvertrieb im Internet.[35] Bei einem proprietären Standard sind darüber hinaus auch freie Betriebssysteme außen vor, was ihre Chancen auf eine Verbreitung insbesondere unter Privatnutzern drastisch reduziert. Es stellt sich also die Frage, ob eine derartige Technologie allein der Unterhaltungsindustrie und ihrer Lobby überlassen werden soll. Es ist dringend geboten, die Agenda um Verbraucherperspektiven zu erweitern, um einer aggressiven Industriepolitik Einhalt zu gebieten.

Nur unter Einbeziehung von Open Source kann es gelingen, transparente Strukturen zu gewährleisten und Verbrauchern zu ermöglichen, statt Restriktionen die eigenen Rechte zu verwalten. Es droht eine einseitige rechtliche und technische Ausgestaltung von DRM. Perspektiven für Open Source im Bereich DRM sind schwer erkennbar. Allerdings kann ausgerechnet die Technologie, die bei *Trusted Computing* (TC) zum Einsatz kommt, hierfür eine Lösung bieten. Das Vertrauen, das bei proprietärer Software aus der Nichtanpassbarkeit resultiert, kann bei TC in die Zertifizierungsprozesse verlagert werden. Dabei muss es allerdings auf offenen Standards und standardisierten

gen eingeladen. Außerdem standen Verbraucheraspekte als letzter Punkt auf der Tagesordnung, konnten schließlich aus Zeitmangel nicht mehr behandelt werden. Sie tauchen im Abschlussreport deshalb nicht auf (Ermert und Kuri 2004).

35 Interessant erscheint in diesem Zusammenhang auch die Idee, Datenschutz mit Hilfe von DRM-Systemen zu ermöglichen (Tóth 2004).

Zertifikaten (wie etwa *common criteria*[36]) aufbauen. Denn es darf nicht übersehen werden, dass Abhängigkeiten von denjenigen entstehen, die die Zertifikate ausstellen. Wenn es möglich sein soll, auch mit Open-Source-Software DRM-geschützte Inhalte abzuspielen, müssen die Anbieter von DRM-Inhalten ihren DRM-Schutz so ausgestalten, dass jede Software, die den Anforderungen an den technischen Schutz genügt, auch ein Zertifikat erhalten kann.

Literaturverzeichnis

Anderson, R. (2004), 'Trusted Computing' Frequently Asked Questions – TC / TCG / LaGrande / NGSCB / Longhorn / Palladium / TCPA', http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html [29. Okt 2004].

Becker, A. und Ziegler, M. (2000), 'Wanted: Ein Überlebensmodell für die Musikindustrie Napster und die Folgen'. White Paper der Diebold Deutschland GmbH, Eschborn.

Berlin Declaration on Collectively Managed Online Rights: Compensation without Control (2004), http://www.wizards-of-os.org/index.php?id=1699 [18. Okt 2004].

Bomb The Bass (1988), 'Beat Dis', Mister-Ron DOOD1.

Bundesministerium der Justiz (BMJ) (2004), 'Referentenentwurf für ein zweites Gesetzt zur Regelung des Urheberrechtes in der Informationsgesellschaft'.

Bundesverband der Phonographischen Wirtschaft e. V. (2004), 'Jahreswirtschaftsbericht 2003'.

Computerwoche (1997), 'Siegen die liberalen Kräfte im Streit ums Kryptogesetz?', *Computerwoche* **14**(14), S. 9–10.

Creative Commons (2004), 'Frequently Asked Questions', http://creativecommons.org/faq [15. Sep 2004].

Drummond, B. und Cauty, J. (1988), *The Manual*, KLF Publications/Ellipsis, London.

Ermert, M. und Kuri, J. (2004), 'Europäische Kommission startet Konsultation zu DRM', heise online http://www.heise.de/newsticker/meldung/49253 [20. Okt 2004].

Fetscherin, M. (2004), 'Digital Rights Management Systeme: Stand der Technik'. http://www.ie.iwi.unibe.ch/staff/fetscherin/resource/2004-01-19-DRM-Stand-der-Technik.pdf [18. Sep 2004].

Fraunhofer-Institute for Integrated Circuits (IIS) / Fraunhofer-Institute for Digital Media Technology (IDMT) / Fraunhofer-Institute for Secure Telecooperation (SIT) (2003), 'Light Weight Digital Rights Management LWDRM®', http://www.lwdrm.com/ [10. Okt 2004].

Free Software Foundation (2004), 'Was ist Freie Software?', http://www.germany.fsfeurope.org/documents/freesoftware.de.html [10. Okt 2004].

Friedrichsen, M. e. A. (2004), *Die Zukunft der Musikindustrie – Alternatives Medienmanagement für das mp3-Zeitalter*, Verlag Reinhard Fischer, München.

36 http://www.commoncriteriaportal.org/

Gebhardt, G. (2003), 'Die Musikwirtschaft im Jahr 2010', http://www.ifpi.de/news/news-312.htm [20. Okt 2004]. Keynote auf der popkomm am 15.8.2003.

Gebhardt, G. (2004), 'Sieben Argumente gegen eine Kulturflatrate', http://www.spiegel.de/netzwelt/politik/0,1518,316837,00.html [10. Okt 2004].

Grassmuck, V. (2004), 'Putting users at the centre achieving an 'information society for all'', http://privatkopie.net/files/privatkopie-bof_on-DRM.pdf [20. Okt 2004].

Günnewig, D. (2002), 'Digital Rights Management Systeme: Ein Helfer in der Not?', http://www.datensicherheit.nrw.de/Daten/WS06122002/guennewig.pdf [30. Sep 2004].

Hansen, S. und Block, A. (2004), 'Musik saugen legal – Die Türen zu den Online-Shops sind aufgestoßen', *c't* **6**, S. 176–187.

Harker, D. (1998), 'It's a jungle sometimes. The music industry, the crises and the state'. http://www2.hu-berlin.de/fpm/texte/harker1.htm [01. Nov 2004].

Herrmannstorfer, M. (2004), 'iTunes Music Store als Vorbild für die Musikindustrie', heise online http://www.heise.de/newsticker/meldung/51741 [18. Okt 2004].

High Level Group (2004), 'High Level Group on Digital Rights Management', http://europa.eu.int/information_society/eeurope/2005/all_about/digital_rights_man/doc/040709_hlg_drm_2nd_meeting_final_report.pdf [20. Okt 2004]. Final Report.

KLF (1989), 'What Time Is Love', KLF Communications KLF004.

Kemmler, Sebastian, e. a. (2004), Das Urheberrecht in der Informationsgesellschaft. Studie zum Meinungsbild zur Novellierung des Urheberrechtsgesetzes. Studentische Projektgruppe Infrarot an der Universität der Künste, Berlin.

Kirchhof, P. (1998), *Die Zukunft der Musikindustrie – Alternatives Medienmanagement für das mp3-Zeitalter*, v. Decker und Müller, Heidelberg.

Kulle, J. (1998), Ökonomie der Musikindustrie: Eine Analyse der körperlichen und unkörperlichen Musikverwertung mit Hilfe von Tonträgern und Netzen, PhD thesis, Universität Hohenheim.

Kuri, J. (2003*a*), 'Musikindustrie sieht noch kein Ende der Talfahrt', heise online http://www.heise.de/newsticker/meldung/39996 [24. Aug 2004].

Kuri, J. (2003*b*), 'US-Gericht verweigert Schließung von Online-Tauschbörsen', heise online http://www.heise.de/newsticker/meldung/36411 [16. Nov 2004].

Kuri, J. (2003*c*), 'Werbeverband hält Kampagne gegen Raubkopierer für äußerst fragwürdig', heise online http://www.heise.de/newsticker/meldung/42578 [24. Aug 2004].

Kuri, J. (2004), 'Musikindustrie will Online-Plattform Phonoline aufgeben [Update]', heise online http://www.heise.de/newsticker/meldung/51482 [28. Sep 2004].

Land Hessen (1946), 'Verfassung des Landes Hessen (HLV)', http://www.hessenrecht.hessen.de/gvbl/gesetze/10_1Verfassung/10-1-verfass/verfass.htm [30. Sep 2004].

Lau, P. (2002), 'Musik für Erwachsene (2)', *brand eins* **9**, S. 51 ff. http://www.brandeins.de/home/inhalt_detail.asp?id=325&MenuID=130&MagID=11&sid=su2179316192852774 [14. Nov 2004].

Lessig, L. (2004), *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*, The Penguin Press, New York, NY. http://www.free-culture.cc/freeculture.pdf [01. Nov 2004].

M/A/R/R/S (1987), 'Pump Up The Volume', 4AD AD70.

Microsoft Deutschland (2004), 'Wie funktioniert DRM?', http://www.microsoft.com/germany/digital-mentality/funktionsweise.mspx [30. Sep 2004].

Netzeitung (2004), 'Fusion von Sony Music und BMG perfekt', *Netzeitung* **6**. http://www.netzeitung.de/wirtschaft/unternehmen/299070.html [15. Sep 2004].

New Order (1983), 'Blue Monday', Factory FAC73.

Oechsler, J. (1998), 'Skript zum Urheberrecht und gewerblichen Rechtsschutz'. http://www.uni-potsdam.de/u/ls_oechsler/lehre/skript/ur.pdf [01. Nov 2004].

Portishead (1994), 'Sour Times', Go Beat GODCD116.

Stallman, R. (2002), 'Can you trust your computer?', *News Forge* . http://www.newsforge.com/business/02/10/21/1449250.shtml?tid=19 [19. Okt 2004].

Stein, S. (2004), 'Drei Jahre Knast für Tauschbörsenbenutzer!', *Computerbild* **20**.

Tóth, G. (2004), 'DRM and privacy – friends or foes?', *INDICARE Monitor* **1**(4). http://indicare.berlecon.de/tiki-read_article.php?articleId=45 [20. Okt 2004].

Wang, X. (2004), 'MPEG-21 Rights Expression Language: Enabling Interoperable Digital Rights Management', http://www.computer.org/multimedia/mpeg.htm [20. Okt 2004].

Wicke, P. (1997), 'Musikindustrie im Überblick – Ein historisch-systematischer Abriß'.

Wilkens, A. (2004), 'Online-Tauschbörsen sind beliebter denn je', heise online http://www.heise.de/newsticker/meldung/49023 [16. Nov 2004].

World Intellectual Property Organization (WIPO) (1996), 'WIPO Performances and Phonograms Treaty, Geneva', http://www.wipo.int/documents/en/diplconf/distrib/95dc.htm [20. Sep 2004].

Das Netlabel als alternativer Ansatz der Musikdistribution

SEBASTIAN REDENZ

Schon lange vor der Entwicklung von Peer-to-Peer-Netzwerken und Bezahldownloads gab es virtuelle Musiklabel, die legal digitale Musikdateien anboten. Unabhängig von rückgängigen Verkaufszahlen in der Musikbranche, arbeiten Netlabels konsequent an einem alternativen Vertriebsmodell für Musik. Wie dies unter rechtlich legalen Rahmenbedingungen bereits seit Jahren geschieht, soll der vorliegende Beitrag verdeutlichen, der die Motivation und Systematik des vom Autor geleiteten Netlabels *Thinner*[1] beschreibt. Hierbei untersucht der Artikel die historische Entwicklung der Netlabels und benennt deren Ursprünge. Unter Berücksichtigung kausaler Zusammenhänge, die zur aktuellen wirtschaftlichen Depression innerhalb der *Independent Musikindustrie* für elektronische Musik führten, werden Gemeinsamkeiten und Unterschiede zwischen konventionellen Musiklabels[2] und Netlabels und daraus resultierende Konsequenzen erörtert, während abschließend Chancen und Risiken letzterer genannt werden.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Das Wissen der Welt – Die Wikipedia

PATRICK DANOWSKI UND JAKOB VOSS

(CC-Lizenz siehe Seite 463)

Freie Inhalte sind auf dem Vormarsch. Kombiniert mit kollaborativem Arbeiten entstehen außerordentliche Projekte, die es in dieser Form bisher noch nicht gegeben hat. Mit an vorderster Front befindet sich das wohl umfangreichste Projekt dieser Sorte – die Internet-Enzyklopädie Wikipedia. Ermöglicht wird sie durch das recht simple „Wiki-Prinzip", nach dem jeder, ohne Anmeldung oder Filterung durch Redakteure, Inhalte erstellen und bearbeiten kann. Tausende Freiwillige arbeiten aktiv an der Gestaltung der Enzyklopädie und setzen dadurch die einzig bestehende Regel des Systems um – den „Neutralen Standpunkt". Die Autoren beschreiben in dem Artikel die Geschichte der Wikipedia und gehen detailliert auf deren Struktur ein. Zudem erklären sie den alltäglichen Ablauf, wie Beiträge geschaffen werden und wie deren Qualität gesichert werden kann. Abschließend geben sie einen Ausblick über die Zukunft der Enzyklopädie sowie weitere spannende und vielversprechende Projekte mit freien Inhalten. Der Beitrag basiert auf einem im Dezember 2004 in der Ausgabe 8/2004 der Fachzeitschrift „Information – Wissenschaft und Praxis"[1] erschienenen Artikel. Er wurde von den Autoren und Erik Möller in einem Wiki unter http://meta.wikimedia.org/wiki/Open-Source-Jahrbuch/Beitrag_2005 überarbeitet und steht unter den Bedingungen der GNU Free Documentation License und der Creative–Commons-Lizenz „Attributions-ShareAlike" zur Verfügung.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Kapitel 7

Open Innovations

„Innovation“ – eine Spurensuche

ROBERT A. GEHRING

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Geistiges Eigentum im Internet: Ist alte Weisheit ewig gültig?

James Bessen und Eric Maskin

Das Wachstum des Internets setzt die etablierten Formen des Schutzes geistigen Eigentums wie Urheber- und Patentrechte unter Druck. Informationen, einmal ins Internet gestellt, lassen sich leicht kopieren. Wo die Kopierkosten niedrig sind und die Anonymität gewahrt bleibt, reagieren die Rechteinhaber mit verschärfter Durchsetzung bestehender Rechtsansprüche und rufen nach dem Gesetzgeber. Doch der Ansatz, bestehende Rechtsansprüche zu verschärfen und neue zu schaffen, erweist sich im Internet als problematisch, wenn es darum geht, Innovation zu fördern. Im vorliegenden Aufsatz wird an Beispielen gezeigt, dass Anwender innovative Beiträge leisten, denen aber Urheberrecht und Patentrecht entgegenstehen. Die Autoren argumentieren, dass das Internet spezifische ökonomische Eigenschaften aufweist, die eine reflexartige Verschärfung von Ansprüchen auf geistiges Eigentum jedenfalls im Internet als ein ungeeignetes Instrument zur Innovationsförderung erscheinen lassen. Vielmehr gibt es empirische Evidenz dafür, dass schwächere Rechte des geistigen Eigentums die sequentielle Innovation besser unterstützen würden: Imitation erhöht in einer dynamischen Umgebung die Innovationsanreize; Lizenzierungskosten vermindern Innovationsanreize; reines Kopieren wäre zu verhindern, kreative Nachahmung hingegen zu fördern.*

1. Einleitung

Das Wachstum des Internets setzt die etablierten Formen des Schutzes geistigen Eigentums wie Urheber- und Patentrechte unter Druck. Informationen, einmal ins Internet gestellt, lassen sich leicht kopieren. Wo die Kopierkosten niedrig sind und die Anonymität gewahrt bleibt, reagieren die Rechteinhaber mit verschärfter Durchsetzung bestehender Rechtsansprüche und rufen nach dem Gesetzgeber, den Rechtsschutz auszuweiten: Zusätzlich zu den bestehenden Schutzrechten sollen neue Formen

* Der Text wurde von den Autoren für das vorliegende Buch aktualisiert. Die Übersetzung erfolgte mit Genehmigung seitens der Autoren durch Bastian Zimmermann, Clemens Brandt und Robert A. Gehring. Wir danken den Autoren für die Genehmigung zum Abdruck.

von „Inhalten", Medien und Zugriffsmöglichkeiten von Gesetzes wegen unter Schutz gestellt werden. Man kann diese Bemühungen als Teil einer seit 20 Jahren währenden Entwicklung sehen, die stets in Richtung Ausweitung von geistigen Eigentumsrechten und Intensivierung von deren Durchsetzung geht. Allerdings ist diese Reaktion nicht notwendigerweise angemessen, zumal im Internet.

In diesem Aufsatz wird argumentiert, dass das Internet und das darauf aufbauende World Wide Web spezifische Eigenschaften aufweisen, die einen solchen Ansatz – die permanente Verschärfung der Schutzrechte – als ungeeignet erscheinen lassen. Das Internet ist eine hochgradig dynamische und interaktive Gemeinschaft. Und tatsächlich ist ein großer Teil der Software, auf der das Web basiert, Open-Source-Software. Der vorliegende Aufsatz präsentiert in aller Kürze ein formales ökonomisches Modell zur Beschreibung einer solchen dynamischen und interaktiven Umgebung. Das Modell legt den Schluss nahe, dass unter den im Internet herrschenden Bedingungen sowohl die einzelnen Rechteinhaber als auch die Gesellschaft im Ganzen eher von *schwächeren* als von verschärften Schutzrechten profitieren würden.

Politiker sollten daher vorsichtig sein: Die überlieferte Ansicht, dass stärkere Schutzrechte für geistiges Eigentum automatisch die Innovationsanreize verstärken würden, basiert auf ökonomischen Modellvorstellungen, die dem Internet oft nicht gerecht werden.

2. Das traditionelle Modell von geistigem Eigentum

Üblicherweise wird strenger Schutz für geistiges Eigentum immer damit begründet, dass er für Autoren und Erfinder den Anreiz bewahrt, überhaupt etwas zu schaffen. Ausführlich lautet die Begründung folgendermaßen:

Kreative Tätigkeit birgt typischerweise erhebliche Kosten. Zwar sind Künstler, Autoren und Erfinder nicht unbedingt ausschließlich oder vorrangig von der Aussicht auf finanziellen Gewinn motiviert; dennoch kann der kreative Prozess, von der ursprünglichen Idee über ihre Entwicklung bis hin zur Verbreitung eines Werkes, so aufwendig sein, dass viele Urheber einen finanziellen Ertrag brauchen, um ihre Entwicklungskosten wieder zu erwirtschaften. Dieser Ertrag dient daher als „Innovationsanreiz".

- Wenn eine Arbeit kopiert wird, entgehen dem eigentlichen Autor/Erfinder mögliche Verkäufe, also Profite. Deshalb reduziert eine Umgebung, die das (kostenlose) Kopieren ermöglicht, den Innovationsanreiz. Mit einer geringeren Aussicht auf Profite werden manche Urheber die anfänglichen Entwicklungsinvestitionen nicht aufbringen können oder wollen. Sie werden sich daher gegen eine kreative Tätigkeit entscheiden.
- Der Rechtsschutz für geistiges Eigentum erschwert die Nachahmung und steuert so diesem Erosionseffekt entgegen. Der Schutz ermutigt Erfinder und Autoren in ihrem Schaffensprozess, wovon im Gegenzug die gesamte Gesellschaft profitiert.

In einem solchen Modell ist stärkerer Schutz automatisch besser: Stärkerer Schutz bewirkt eine Abnahme der Imitation, erhöht somit die Investitionsanreize und führt

letztendlich zu höherer Wohlfahrt. Diese Argumentation klingt überzeugend und ist seit über 200 Jahren grundlegend für einen starken Schutz des geistigen Eigentums.

Und doch ist das ökonomische Modell, das diesem traditionellen Argument zugrunde liegt, überraschend beschränkt. In Wahrheit ist kreative Tätigkeit oft nicht der Arbeit einsamer Schöpfer geschuldet. Vielmehr ist sie interaktiv und schließt Beiträge von vielen verschiedenen Seiten ein. Tatsächlich findet Innovation oft schrittweise statt. Jeder Innovationsprozess baut auf den Ergebnissen des vorangegangenen Schrittes auf, um eine Verbesserung zu erzielen. Das Standardmodell setzt Nachahmung oft mit Kopieren gleich. Wenn jedoch Innovation schrittweise stattfindet, ist Nachahmen mehr als Kopieren und stellt eine Bereicherung dar.

Das herkömmliche Modell basiert auf der Annahme eines einzelnen Schöpfers. Tief verankert in unserer Kultur ist das Bild des Kreativen als eines romantischen Individuums: der Künstler in der Dachstube oder der Erfinder in der Garage. Ein Teil der Überzeugungskraft des Standardmodells beruht auf unserer Angewohnheit, kreative Tätigkeit für das Spezialgebiet einsamer Genies zu halten.

Ein Ort, wo dieses vertraute Denken mit der Realität in Konflikt gerät, ist das World Wide Web. Das Web wird oft als Gemeinschaft bezeichnet. Es bietet den einzelnen Kreativen eine wunderbare Möglichkeit, ihre Werke zu veröffentlichen; zugleich offeriert es Möglichkeiten zur interaktiven Kommunikation. So bildet es den Nährboden für schrittweise Weiterentwicklungen.

Es ist hilfreich, einige Beispiele interaktiver und schrittweiser Innovationen im Web zu betrachten, um die Unzweckmäßigkeit des herkömmlichen Modells von geistigem Eigentum in dieser Umgebung besser zu verstehen.

3. Einige Beispiele für interaktive und schrittweise Innovation

3.1. Interaktive Foren

Ein bekanntes Beispiel für interaktive Entwicklung ist das *interaktive Forum*. Als Druckzeitschriften zunehmend im Netz veröffentlicht wurden, richteten ihre Herausgeber häufig auch sogenannte Foren ein. Diese interaktiven Webseiten bieten den Lesern bzw. der gesamten Öffentlichkeit die Möglichkeit, unabhängig Kommentare und Meinungen zu publizieren. Typischerweise kommen vielfältige Dialoge mit den Autoren von Artikeln zustande, die auch abgedruckt werden. Manche von ihnen nehmen die Form von Echtzeit-Gesprächen an, andere werden in E-Mail-Archiven gespeichert. Die Autoren steuern neues Material bei und diskutieren darüber mit den Lesern; die Leser wiederum liefern Feedback und bauen die Diskussion oft aus. Im Ergebnis entsteht eine stark erweiterte Version des Leserbriefs – mit sehr viel komplizierteren geistigen Eigentumsrechten.

Eine interaktive Website, auf der der Austausch „Unbedarfter“[1] zu Konflikten mit dem geistigen Eigentum geführt hat, ist das *Online Guitar Archive* (OLGA). OLGA wurde 1992 von James Bender gegründet und bietet ein Archiv mit etwa 28 000 von Nutzern eingestellten Gitarren-Tablaturen sowie Gitarrenübungen und andere

1 Das ist in Bezug auf das geistige Eigentum gemeint; Anm. d. Ü.

Unterstützung für Gitarristen. Die ehemals von der University of Nevada, Las Vegas, (UNLV) gehostete Seite war sehr beliebt: Nutzer luden sich von ihr etwa 200 000 Dateien pro Woche herunter.

Gitarren-Tablaturen sind eine Form der Musiknotation, die Bund- und Saiten-Fingersätze anzeigen. Die Tablaturen werden üblicherweise von Liedtexten begleitet. Da Gitarrenakkorde oft auf verschiedene Art gegriffen werden können, bieten die Tablaturen Anleitungen zu ihrer Umsetzung. Insbesondere helfen sie Gitarristen, die wie bestimmte Künstler auf CDs und anderen Aufnahmen klingen wollen. Für deren Spiel sind oft keine Noten verfügbar und selbst wenn, so treffen diese im Allgemeinen nicht genau auf die Aufführung auf der Aufnahme zu. Tablaturen werden üblicherweise von anderen Musikern als den ursprünglich spielenden notiert. Zwar werden sie mithilfe von Aufnahmen nach Gehör aufgeschrieben, aber verschiedene Musiker nehmen dieselbe Aufnahme unterschiedlich wahr, und kommerzielle Notenblätter neigen gern dazu, vereinfachte Fingersätze mit einem „hübschen" Klang zu präsentieren.

Die auf OLGA veröffentlichten Tablaturen stellen individuelle Interpretationen einzelner Gitarristen dar, die anhand von Aufnahmen ihre eigenen Fingersätze nieder geschrieben haben. Durch ihre Individualität fügen sie sowohl Aufnahmen als auch Notenblättern (so vorhanden) einen Mehrwert hinzu. Dessen ökonomische Bedeutung besteht darin, dass die Tablaturen eine Ergänzung zu den Audioaufnahmen darstellen, keinen Ersatz. Beide, der Verfasser der Tablatur und ihr Nutzer, der ein Lied einüben will, werden mit dem Anhören einer Aufnahme beginnen. Zwar mögen die Gitarren-Tablaturen gelegentlich Notenblätter ersetzen, aber insgesamt betrachtet stellen sie eine ökonomisch wertvolle Ergänzung zu Aufnahmen dar.

Dogmatisch betrachtet, stellt die Verbreitung von Gitarren-Tablaturen über das Internet wohl eine Verletzung traditioneller geistiger Eigentumsrechte dar. Die Plattenfirma EMI jedenfalls sah das so und schickte im Januar 1996 der UNLV einen Brief, in dem sie rechtliche Schritte androhte. Die UNLV schloss OLGA umgehend. Zahlreiche gespiegelte Server wurden ebenfalls vom Netz genommen. Einige gespiegelte Server blieben am Netz, jedoch mit verringerten Möglichkeiten, neue Tablaturen beizusteuern. Etliche Gitarristen im Web waren empört und begannen damit, EMI–Aufnahmen zu boykottieren. Für OLGA selbst – ohne rechtlichen Stand – war es nicht möglich, EMI zu Gesprächen über eine Verhandlungslösung zu bewegen. Im Jahr 1998 verschärfte sich die Situation noch, als die „Harry-Fox-Agentur" ins Spiel kam. Bei der Harry-Fox-Agentur handelt es sich um eine Organisation, die Musikaufnahmen lizenziert. Ohne Angabe konkreter Titel, die Urheberrechte verletzen würden, gab die Harry-Fox-Angentur dem OLGA sieben Tage Zeit, das komplette Archiv einschließlich gespiegelter Server zu schließen. Gespräche über mögliche Lizenzierungen wurden verweigert. OLGA konnte nur weiterbestehen, indem ein neues Archiv aufgebaut wurde, in das nur noch Gitarren-Akkordtabellen ohne komplette Liedtexte aufgenommen werden.

EMIs Handeln mag im Kontext der traditionellen Urheberrechtsvorstellung sinnvoll gewesen sein. In der Welt des Webs aber, in der Nutzer Mehrwert beisteuern, erscheint EMI kurzsichtig. Aus unserer Perspektive wäre es für EMI besser gewesen, das Hosting für die OLGA-Site zu übernehmen und die dort gespeicherten Tablaturen

mit dem eigenen Bestand an Aufnahmen, Fanclub-Informationen usw. zu verknüpfen. Andere Unternehmen geben in der Tat eine Menge Geld aus, um genau solche Dinge zu tun. Was wäre geschickter, als die Schirmherrschaft über eine ohnehin schon erfolgreiche Website zu übernehmen?

Im traditionellen Modell des geistigen Eigentums läßt sich die Wertschöpfung Außenstehender nicht integrieren. Damit eignet es sich nur schlecht als Leitfaden für Unternehmen oder Politik.

3.2. Sequentielle Entwicklung

Der Bereich der Publikation von Software bietet ein besonders anschauliches Beispiel für sequentielle Entwicklung, denn der Rechtsschutz für Software durchlief in den 80er Jahren eine Art natürliches ökonomisches Experiment. Zwar bildet die Verbreitung von Software im Web einen eigenständigen Bereich der Publikation, aber andere Publikationswege sind von Entwicklungen beim Schutz des geistigen Eigentums an Software in zweierlei Hinsicht betroffen.

Zum einen werden alltägliche Webaktivitäten von Patentinhabern potentiell bedroht, die ihre Rechte ausüben wollen könnten. Regelmäßig tauchen derartige Patente mit breiten Auswirkungen auf im Web übliche Handlungen auf, wie z. B. Comptons Multimedia-Patent, das Freeny-Patent auf elektronische Verkäufe und Unisys' GIF-Patent.

Zum anderen werden Inhalte häufig gemeinsam mit Software angeboten, wobei die Software den Inhalt überhaupt erst zugänglich macht. Beispiele dafür sind aktuelle Nachrichten, die zusätzlich zu den traditionellen Verbreitungsformen nun in Form von Datenbanken, Faxdiensten, E-Mail-Angeboten, Echtzeit-Meldungen und Radioberichten angeboten werden. Datenbankanbieter liefern oft auch einen Service, der an bestimmte Zugriffsarten gebunden ist, und heute stehen Technologien zur Verfügung, die einen technisch kontrollierten Zugang zu so ziemlich jedem Inhalt ermöglichen.

Die besondere Bedeutung von Software für unsere Diskussion liegt nun darin, dass Software typischerweise einer schnellen und schrittweisen Entwicklung unterworfen ist: Jede Innovation wird von Wettbewerbern imitiert und verbessert, wobei jeder einen einzigartigen Beitrag leistet. Softwarekonzepte, die in den 60er Jahren konzipiert wurden, wie das elektronische Publizieren, der Hypertext, Multimedia und künstliche Intelligenz, mussten erst über Jahrzehnte von vielen Firmen immer wieder verbessert werden, bevor sie kommerziellen Erfolg auf breiter Front hatten.

Die Veränderungen des Rechtsschutzes für Software starteten Mitte der 80er Jahre ein ökonomisches Experiment. Bis zu diesem Zeitpunkt war Software weitgehend durch das Urheberrecht geschützt und eine Reihe von Prozessen hatte dazu beigetragen, den Schutzumfang zu präzisieren. Es war klar, dass unmittelbares Kopieren illegal war, andere Formen der Nachahmung aber nicht, solange sie bei gleicher Funktionalität anders verwirklicht worden waren. Patente auf Softwareerfindungen wurden bis dahin nur ausnahmsweise von Gerichten bestätigt.

Ab Mitte der 80er und während der 90er Jahre wurde der Patentschutz für Software durch Gerichtsurteile erheblich ausgeweitet. Im Ergebnis gab es geradezu eine

Explosion der jährlich erteilen Softwarepatente auf über 20 000 (vgl. Abb. 1). Damit erreichten Softwarepatente einen Anteil von mehr als 15 % aller Patente (vgl. Abb. 2). Bis dato wurden in den USA mehr als 200 000 Softwarepatente erteilt. Dem traditionellen ökonomischen Modell zufolge hätte dieser entscheidend erweiterte Rechtsschutz für geistiges Eigentum die Innovationsanreize deutlich erhöhen müssen. Firmen, die aufgrund des Nachahmungsrisikos bisher davor zurückgeschreckt waren, hätten mit neuen Produkten in den Markt eintreten sollen. Projekte, deren Finanzierung bisher zu riskant erschien, hätten nunmehr realisierbar werden müssen. Es hätten in der Folge die Forschungs- und Entwicklungsausgaben steigen müssen.

Vielen aus der Software- und Computerbranche erschien die Ausweitung des Patentschutzes auf Software als ein Versuch, etwas zu reparieren, das gar nicht kaputt war. Diese Industriezweige waren doch bereits hochinnovativ: Neugründungen von Firmen, Produktinnovationen sowie Ausgaben für Forschung und Entwicklung waren im Verhältnis zu den Verkäufen zahlenmäßig hoch. Dennoch stand das hohe Innovationspotential nicht unbedingt im Widerspruch zum traditionellen Modell: Die innovativen Firmen hätten sozusagen bloß die Spitze eines Eisberges gebildet, wobei sie ausschließlich die profitabelsten Innovationen realisiert hätten, um in Anbetracht des Risikos der Nachahmung bestehen zu können.

Folgt man dieser Art der Argumentation, hätte ein stärkerer Rechtsschutz für geistiges Eigentum zu einem noch höheren Niveau an innovativer Aktivität führen müssen. Darüber hinaus hätte man erwarten können, dass stärkere Anreize mehr innovativen Start-up-Firmen den Eintritt in die jeweilige Branche erleichtern würden. In der Tat waren kleine Start-up-Firmen in der Vergangenheit eine wichtige Innovationsquelle in der Softwareindustrie.

Doch es geschah nichts dergleichen. Eine Untersuchung von Bessen und Hunt (2004) belegt detailliert, dass Softwarepatente größtenteils von großen, etablierten Unternehmen erworben wurden. Die meisten von ihnen sind im Hardwarebereich aktiv. Mögen sie auch Software in ihre Produkte integrieren (z. B. in Kopiergeräte), stellt sie dennoch nicht ihr eigentliches Produkt dar. Die kleinen Start-up-Firmen, die für die Innovation in der Softwareindustrie so wichtig waren, haben in der Wirklichkeit nur einen sehr geringen „Hang zu Patenten". Daher können Softwarepatente diesen Firmen keine stärkeren Innovationsanreize geboten haben. Noch wichtiger ist, dass Bessen und Hunt (2004) zu dem Schluss kommen, dass diejenigen Firmen, die während der 90er vergleichsweise viele Softwarepatente erhielten, ihre Forschungs- und Entwicklungsausgaben im Vergleich zu den Verkaufszahlen sogar reduzierten.

Es ist klar, dass diese Resultate nur schwer mit dem traditionellen Modell in Einklang zu bringen sind. Es scheint ganz so, dass (a.) Softwarepatente weitgehend auf der Basis von Forschung und Entwicklung erlangt wurden, die sowieso stattgefunden hätten, und dass (b.) die Erweiterung des Patentschutzes sowohl auf Forschungs- und Entwicklungsausgaben als auch auf Firmenneugründungen keinen sehr positiven Einfluss hatte. Nachahmungen scheinen Innovationen in einer dynamischen Umgebung mit rapider und schrittweiser Innovation nicht zu verhindern. Hier spielen Faktoren eine Rolle, die über das traditionelle Modell hinausgehen.

4. Eine kurze ökonomische Analyse

Ein passenderes ökonomisches Modell muss Folgendes berücksichtigen:

1. Kreative Nachahmung unterscheidet sich vom Kopieren; Nachahmer können für wichtige Bereicherungen sorgen.

2. Einige Umgebungen sind statisch, andere dagegen sind von hoher Dynamik und schrittweiser Innovation gekennzeichnet. Geistiges Eigentum kann in jeder dieser Umgebungen eine sehr unterschiedliche Rolle spielen.

3. Manche kreative Arbeiten haben einzelne Autoren, andere haben mehrere, aufeinander folgende.

4. Der Beitrag zusätzlicher Autoren ist oft nicht vorherzusagen, und der Wert jedes Beitrags besteht oft in der Beisteuerung des spezifischen Wissens des jeweiligen Autors.

Ein solches formales Modell zu entwickeln, geht über den Rahmen dieses Aufsatzes hinaus. Jedoch lassen sich einige bedeutende Ergebnisse eines solchen Modells, wie es an anderer Stelle dargestellt wurde (Bessen und Maskin 2000), folgendermaßen zusammenfassen:

1. Nachahmung kann in einer dynamischen Umgebung die Gesamtanreize erhöhen, etwas Neues zu schaffen. Der Großteil kreativer Tätigkeit stellt sich in der Tat als teilweise Nachahmung dar.

2. Starker Rechtsschutz für geistiges Eigentum kann Schaffensanreize dadurch vermindern, dass er Lizenzierungen und andere Formen gemeinsamer Informationsnutzung reduziert.

3. Im Großen und Ganzen ist ein gemäßigter Schutz geistigen Eigentums optimal. Die beste Art von Rechtsschutz für geistiges Eigentum ist stark genug, um unmittelbares Kopieren und geschäftszerstörende Plagiate zu verhindern, aber auch schwach genug, um die wechselseitige Lizenzvergabe zu maximieren und zur intensiven gemeinsamen Informationsnutzung zwischen Wettbewerbern zu ermutigen.

Die ersten drei Jahrzehnte der Halbleiterindustrie geben ein Beispiel dafür, wie Schutz für geistiges Eigentum in einer dynamischen Umgebung gut funktioniert. Angefangen mit den „Bell Labs", die ihre einfachen Patente auf den Transistor an jeden Interessenten gegen eine geringe Gebühr lizenzierten – obwohl es sich dabei um einen möglichen Wettbewerber handeln konnte –, lizenzierten Halbleiterunternehmen allgemein ganze Patentportfolios. Patente verhinderten nicht, dass neue Unternehmen den Markt betraten, und Unternehmen teilten sich wichtige Patente. Dieses Umfeld veränderte sich schließlich mit der in den frühen 80er Jahren vollzogenen, erheblichen Verschärfung des Rechtsschutzes für geistiges Eigentum.

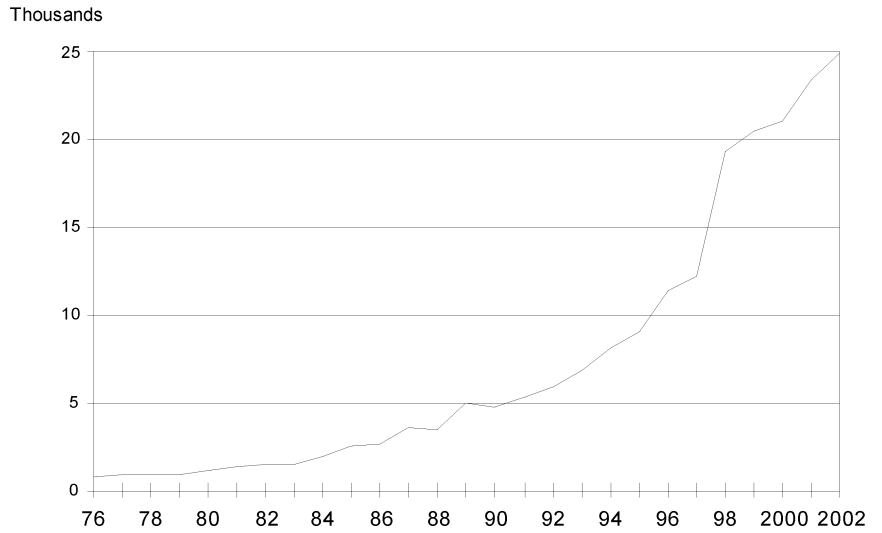

Abbildung 1: Gewährte Softwarepatente in 1 000 (Das Datum bezieht sich auf die Erteilung, nicht die Beantragung des Patents)

Freie und Open-Source-Software sind ebenfalls von dieser Dynamik sequentieller und komplementärer Innovation erfasst. Freie und Open-Source-Lizenzen ermöglichen Imitation und komplementäre Erweiterungen. Einige der Lizenzen verlangen, solche Weiterentwicklungen zu teilen, andere ermutigen lediglich dazu. Im Kern ist es aber das Teilen, das die Dynamik der Entwicklungen bestimmt.

5. Zusammenfassung

Wer Informationen veröffentlicht, ist gut beraten, den Wert der Beiträge anderer zu erkennen und sie zu solchen zu ermutigen. Die Gesellschaft als Ganzes würde davon profitieren anzuerkennen, dass neue Generationen von Autoren bedeutende Beiträge zu vorhandenen Werken leisten. Der Schutz des geistigen Eigentums sollte auf das Maß beschränkt werden, das notwendig ist, um den Ausgleich zwischen der Verhinderung des einfachen Kopierens und der Förderung schöpferischen Imitierens herzustellen.

Gelegentlich wird die Politik des geistigen Eigentums als ein Balanceakt beschrieben, bei dem es darum geht, den Schutz der Anreize zur Innovation gegen die gesellschaftlichen Vorteile einer Weiterverbreitung neuer Ideen abzuwägen. Die hier vorgelegte Analyse macht deutlich, dass sich die Politik nicht nur allgemein der (etwas strukturlosen) Verbreitung von Ideen als solcher widmen sollte, sondern auch den konkreten Formen der Imitation zum Zwecke der Verbesserung, wie sie insbesondere

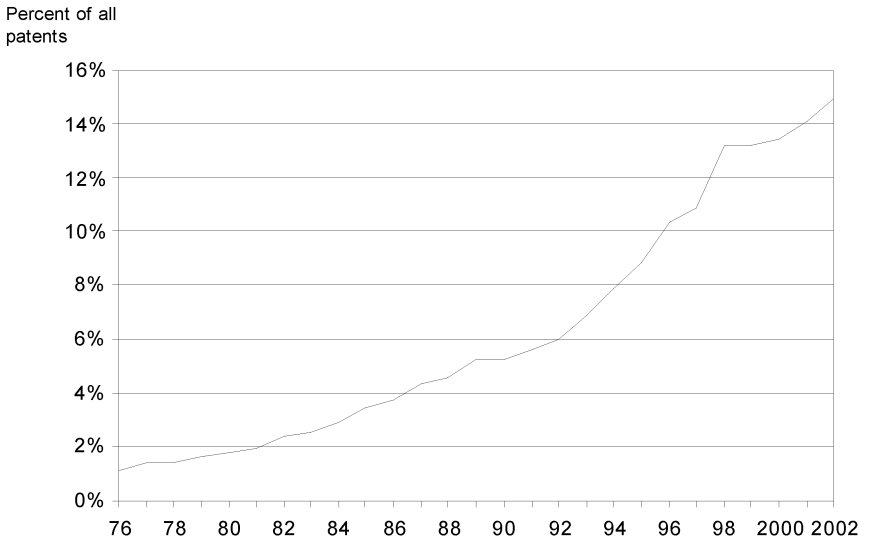

Abbildung 2: Softwarepatente in % aller Patente (Das Datum bezieht sich auf die Erteilung, nicht die Beantragung des Patents)

in dynamischen Umgebungen vorzufinden sind.

Gerade auch das Internet stellt eine hochgradig dynamische Umgebung dar, in der Innovation schrittweise erfolgt. Versuche, dem Internet neue Schutzrechte für geistiges Eigentum aufzuzwingen oder bestehende Schutzrechte zu erweitern, könnten dort unzweckmäßig sein, wo sie den Wert schöpferischer Nachahmung nicht berücksichtigen.

Wie Robert Frost in *„Mending Wall"* schrieb:

> „Bevor ich eine Mauer errichtete, würde ich fragen
> was es wäre, das ich einsperrte oder aussperrte."

Literaturverzeichnis

Bessen, J. und Hunt, R. M. (2004), 'The Software Patent Experiment', Paris. Erscheint demnächst.

Bessen, J. und Maskin, E. (2000), 'Sequential Innovation, Patents and Imitation', MIT Working Paper 00-01, http://www.researchoninnovation.org/patent.pdf [04. Feb 2005].

Das wissenschaftliche Publikationswesen auf dem Weg zu *Open Access*

SÖREN WURCH

Das wissenschaftliche Publikationswesen wird seit Jahrzehnten von einer Krise heimgesucht – der Bibliothekenkrise. Verursacht wurde diese Krise vor allem durch extreme Preissteigerungen bei Abonnements für wissenschaftliche Fachzeitschriften. In der Folge sind viele Bibliotheken nicht mehr in der Lage, ihren Lesern einen umfassenden Bestand dieser Fachzeitschriften zur Verfügung zu stellen. Die wissenschaftliche Kommunikation basiert in erster Linie auf Fachzeitschriften und wird durch deren mangelnde Verfügbarkeit stark eingeschränkt. Kommerzielle Fachverlage wollen mit ihren herkömmlichen Produktionsmethoden nicht zur Lösung dieser Krise beitragen. Daher muss ein neues Modell geschaffen werden, das auch eine strukturelle Veränderung im wissenschaftlichen Publikationswesen mit sich bringt. Ein viel versprechender Ansatz ist das Open-Access-Modell, welches zunehmend Erfolg hat. Damit wird ein freier Zugang zu wissenschaftlichen Publikationen gewährleistet, wie in diesem Artikel beschrieben wird.

„Anwender-Innovationsnetzwerke“: Hersteller entbehrlich

ERIC VON HIPPEL

(CC-Lizenz siehe Seite 463)

Anwender-Innovationsnetzwerke ermöglichen die Entwicklung, Verbreitung und zum Teil auch Produktion von Innovationen – durch Anwender, für Anwender. Klassische Hersteller, der traditionellen Theorie zufolge Quelle von Innovationen, verlieren in solchen Netzwerken an Bedeutung, werden zum Teil gar entbehrlich. Zwei Beispiele für solche horizontal aufgebauten Anwender-Innovationsnetzwerke werden im Folgenden beschrieben: die Community der Entwickler im Bereich von freier und Open-Source-Software sowie die Community der Hochleistungswindsurfer. Es wird diskutiert, welche Voraussetzungen Anwender-Innovationsnetzwerke haben, und unter welchen Bedingungen sie prosperieren. Von Bedeutung sind die Existenz von Innovationsanreizen und Offenlegungsanreizen, wobei dem Verzicht auf Ansprüche aus geistigem Eigentum eine Schlüsselrolle zukommt. Der wesentliche Vorteil der Innovation in diesen Netzwerken ist – aus Anwendersicht – darin zu sehen, dass Innovationen entwickelt werden, die den Bedürfnissen der Anwender genau entsprechen. Dabei werden vorhandene Ressourcen effizienter genutzt – u. a. durch Vermeidung von „Agency-Kosten“ – als in der klassischen, vertikalen Organisation von Innovation, Produktion, Distribution und Nutzung.*

* Die Übersetzung des Textes und die Zusammenfassung stammen von Robert A. Gehring. Wir danken dem Autor, Eric von Hippel, für die freundliche Genehmigung zur Übersetzung und zum Abdruck des vorliegenden Aufsatzes. Das englische Original erscheint in J. Feller, B. Fitzgerald, S. Hissam und K. R. Lakhani, Hrsg. (2005): Perspectives on Free and Open Source Software, The MIT Press, Cambridge, MA.

Diesen interessanten Artikel finden Sie im Open Source Jahrbuch 2005. Auf http://www.opensourcejahrbuch.de gibt es den kostenlosen Download des gesamten Jahrbuches, aller Artikel einzeln und eine Bestellmöglichkeit des gedruckten Buches, das zusätzlich ein Glossar und ein Stichwortverzeichnis enthält. Sie lesen gerade das Open Source Jahrbuch 2005 *light*, kommerzielle Nutzung und Vertrieb sind erlaubt.

Lizenzen

Creative Commons

- Sie dürfen den Inhalt vervielfältigen, verbreiten und öffentlich aufführen.
- Nach Festlegung des Rechtsinhabers können die folgenden Bedingungen gelten:

  
  Namensnennung (by) Sie müssen den Namen des Autors/Rechtsinhabers nennen.

  
  Keine kommerzielle Nutzung (nc) Dieser Inhalt darf nicht für kommerzielle Zwecke verwendet werden.

  
  Weitergabe unter gleichen Bedingungen (sa) Wenn Sie diesen Inhalt bearbeiten oder in anderer Weise umgestalten, verändern oder als Grundlage für einen anderen Inhalt verwenden, dann dürfen Sie den neu entstandenen Inhalt nur unter Verwendung identischer Lizenzbedingungen weitergeben.

  
  Keine Bearbeitung (nd) Der Inhalt darf nicht bearbeitet oder in anderer Weise verändert werden.

- Im Falle einer Verbreitung müssen Sie anderen die Lizenzbedingungen, unter die dieser Inhalt fällt, mitteilen.
- Jede dieser Bedingungen kann nach schriftlicher Einwilligung des Rechtsinhabers aufgehoben werden.
- Die gesetzlichen Schranken des Urheberrechts bleiben hiervon unberührt.
- Nachfolgend die CC-Lizenzen der Version 2.0:

  Attribution (by)
  http://creativecommons.org/licenses/by/2.0/

  Attribution-NoDerivs (by-nd)
  http://creativecommons.org/licenses/by-nd/2.0/

  Attribution-NonCommercial-NoDerivs (by-nc-nd)
  http://creativecommons.org/licenses/by-nc-nd/2.0/

  Attribution-NonCommercial (by-nc)
  http://creativecommons.org/licenses/by-nc/2.0/

  Attribution-NonCommercial-ShareAlike (by-nc-sa)
  http://creativecommons.org/licenses/by-nc-sa/2.0/

  Attribution-ShareAlike (by-sa)
  http://creativecommons.org/licenses/by-sa/2.0/

  Eine Übersicht befindet sich auch unter http://creativecommons.org/licenses/.

The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

End of Terms and Conditions

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the program's name and a brief idea of what it does.
Copyright (C) yyyy name of author

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands show w and show c should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than show w and show c; they could even be mouse-clicks or menu items—whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program
> 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

The GNU Free Documentation License

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.
This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.
We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.
A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.
A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.
The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.
The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.
A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.
Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.
The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.
A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.
The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . .Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Mitwirkende

Matthias Bärwolff (M. A.) ist wissenschaftlicher Mitarbeiter im Fachgebiet Informatik und Gesellschaft der Technischen Universität Berlin und hat das Kapitel „Ökonomie“ betreut.

Andreas Bauer ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Software & Systems Engineering von Prof. Broy an der Technischen Universität München. Er ist selbst langjähriger Autor freier Software und hat in verschiedenen Fachzeitschriften Artikel über deren besondere Anwendungsmerkmale und Entwicklung geschrieben. Weiterführende Informationen über Projekte und Veröffentlichungen finden sich auf seiner Homepage unter http://home.in.tum.de/baueran/.

James Bessen lehrt Recht an der Boston University School of Law. Als ein früherer Innovator schrieb er eines der ersten Programme für das Desktop-Publishing, war CEO eines Softwareunternehmens und erforscht gegenwärtig die Ökonomie der Innovation. James Bessen gibt den „Technological Innovation and Intellectual Property“-Newsletter heraus (http://www.researchoninnovation.org/tiip/index.htm).

Clemens Brandt studiert Informatik seit Oktober 2000 an der Universität Hamburg und an der Technischen Universität Berlin. Derzeitig nimmt er an einem Austauschjahr mit der McGill University in Montréal, Kanada teil. Sein Schwerpunkt im Studium ist Informatik und Gesellschaft und er hat für das Jahrbuch das Kapitel „Open Content“ betreut.

Nicole Bräuer (M. A.) arbeitet derzeit bei DaimlerChrysler im Bereich Markenkommunikation. Wir konnten sie dafür gewinnen, das Lektorat für weite Teile des Buches durchzuführen.

Horst Bräuner ist seit 1989 IT-Leiter bei der Stadt Schwäbisch Hall. Zudem ist er Geschäftsführer der tosipa-SHA GmbH. Zuvor absolvierte er die Ausbildung zum Diplom Verw. Wirt (FH). Er besuchte verschiedenste Seminare im IT-Umfeld u. a. zum *Certified Security Engineer*. Des weiteren ist Horst Bräuner als Projektleiter, Dozent und Internet-/Netzwerk-Verantwortlicher tätig.

Carsten Brunke (Dipl. Inform.) ist Gründer und Geschäftsführer der inmedias.it GmbH (http://www.inmedias.it). Nach seinem Studium an der FH Gießen-Friedberg arbeitete er zunächst in der IT-Abteilung eines großen Medienkonzerns, um sich dann 1995 als IT-Consultant selbstständig zu machen. Mit der 1998 erfolgten Gründung der inmedias.it GmbH in Hamburg hat sich sein Wirken auf Freie Software fokussiert. So ist Carsten Brunke heute im Arbeitskreis Open Source der Hamburger Initiative Hamburg@Work und im Linux-Verband e. V. (LIVE) aktiv. Mit seinem Unternehmen tritt er als Träger der Free Software Foundation Europe (http://www.fsfe.org) auf.

Patrick Danowski ist Dipl. Informatiker und seit Oktober 2004 Bibliothekreferendar an der Zentral- und Landesbibliothek Berlin. Er beschäftigt sich mit Digitalisaten und dem elektronischen Publizieren.

Thomas Ebinger (LL. M. Rechtsinformatik) arbeitet als Rechtsanwalt in Berlin insbesondere im IT-Recht und ist Partner der Kanzlei Berger Groß Höhmann. Er hat bereits mehrfach zu rechtlichen Fragen von Open-Source-Software veröffentlicht; nach Studium und Referendariat in Berlin erwarb er mit Zusatzstudium der Rechtsinformatik/IT-Law in Hannover und London (http://www.eulisp.de) seinen Master of Laws (LL. M.) 2001 mit einer Abschlussarbeit über Open-Source-Software und Softwarepatente; 2001 bis 2003 arbeitete er als Rechtsanwalt in München in einer renommierten internationalen Kanzlei im IT-Recht.

Oliver Feiler hat Abitur, ist selbständig und arbeitet in einer kleinen Firma, die nach einem Richtungswechsel jetzt Software für den Apple Macintosh entwickelt. Sämtliche Programmierkentnisse stammen aus Eigeninitiative. Sein größtes Interesse gilt aber nach wie vor der Entwicklung von Computerspielen und nebenbei der Fotografie. Von ihm stammt der in diesem Buch abgedruckte Sourcecode. Neben dem POP3-Server Snowbox hat er zahlreiche weitere nützliche Programme geschrieben und ist auch in andere Open-Source-Projekte involviert.

Robert A. Gehring studierte Elektrotechnik, Informatik und Philosophie an Technischen Hochschule Ilmenau und an der Technischen Universität Berlin. Nach dem Studium arbeitete er freiberuflich als Consultant, Dozent und Autor, bevor er an die Technischen Universität Berlin zurückkehrte. Dort arbeitete er im Fachgebiet Informatik und Gesellschaft als wissenschaftlicher Mitarbeiter mit den Forschungsschwerpunkten Open Source, IT-Sicherheit und „geistiges Eigentum". Er promoviert zu Fragen der Softwareökonomie und des Softwarerechts. Neben der Wahrnehmung der Aufgaben als Herausgeber hat Robert Gehring für dieses Jahrbuch das Kapitel „Open Innovations" betreut.

Christian F. Görlich (Studiendirektor) machte an der Universität in Münster sein Staatsexamen in den Fächern Deutsch, Philosophie, Soziologie und Erziehungswissenschaften. Er ist Leiter des Seminars Gymnasium/Gesamtschule im Studienseminar für Lehrämter an Schulen in Hamm. Nach einer zeitweiligen Abordnung an das Schulkollegium in Münster übernahm er in der Lehrerausbildung die Fachleitung für die Fächer Sozialwissenschaften und Philosophie am Studienseminar für die Sekundarstufe II in Hamm. Er ist weiterhin wissenschaftlicher Mitarbeiter (Zentrum für interdisziplinäre Forschung, Bielefeld) im Forschungsprojekt „Theoriebildung als Gruppenprozeß" (Zuständigkeitsbereich Psychoanalyse) und ist Lehrbeauftragter für Philosophie an den Universitäten in Münster und Paderborn.

Annika Held studiert seit 2004 Medienberatung an der Technischen Universität Berlin. Zuvor hat sie einen Bachelor of Arts in Philosophy and Economics an der Universität Bayreuth absolviert und ihre Abschlussarbeit über Open-Source-Software

als Herausforderung des Rechts „geistigen Eigentums" geschrieben. Für dieses Buch hat sie Marketingaufgaben übernommen.

Joachim Henkel (Prof. Dr.) ist Inhaber des Dr. Theo Schöller Stiftungslehrstuhls für Technologie- und Innovationsmanagement an der Technischen Universität München. Er studierte in Bochum und Bonn, promovierte am Graduiertenkolleg „Allokation auf Finanz- und Gütermärkten" der Universität Mannheim und habilitierte sich 2004 an der Ludwig-Maximilians-Universität München mit einer Arbeit zu Open-Source–Aktivitäten von Unternehmen. Er ist verheiratet und hat zwei Kinder.

Jens Herrmann studierte Rechtswissenschaften und Informatik in Berlin. Er beendete sein Studium an der Technischen Universität Berlin mit einer Diplomarbeit zum Thema Vorratsdatenspeicherung im Internet. Zur Zeit arbeitet er zu den Themen Datenschutz, Urheberrecht in der Informationsgesellschaft sowie Freier Software.

Maik Hetmank (Diplom Volkswirt) studierte bis 2004 an der Universität Duisburg-Essen. Die Schwerpunkte seines Studium setzte er in den Bereichen der Sozialen Sicherung und der Besteuerung. Nebenher interessiert er sich schon seit frühester Kindheit für Computer. Seit einigen Jahren befasst er sich auch intensiv mit Open–Source-Software und hat über dieses Thema seine Diplomarbeit verfasst.

Ludger Humbert (Dr. rer. nat.) studierte bis 1976 Informatik an der Universität Paderborn. Nach seinem zweiten Studium wurde er Informatiklehrer. An der Universität Siegen promovierte er 2003 in der Didaktik der Informatik mit dem Thema „Zur wissenschaftlichen Fundierung der Schulinformatik". Derzeit ist er Informatiklehrer an der Willy-Brandt-Gesamtschule in Bergkamen und Informatikdidaktiker und Lehrerbildner am Studienseminar Hamm. Er beschäftigt sich u. a. mit wissenschaftstheoretischen Fragen und Standards zur informatischen Bildung. Darüber hinaus arbeitet er im Aufgabenausschuss für den Bundeswettbewerb Informatik.

Christian Imhorst studierte bis 2004 Politische Wissenschaft und Philosophie an der Universität Hannover. Seine Magisterarbeit schrieb er über „Richard Stallman und die freie Software-Bewegung", die Anfang 2005 unter dem Titel „Die Anarchie der Hacker" als Buch erschien.

Stefan Krempl (Dr.) Jahrgang 1969, arbeitet als freier Autor in Berlin und als Dozent am Südosteuropäischen Medienzentrum in Sofia. Er publiziert regelmäßig in Magazinen und Online-Diensten wie c't, heise online und Telepolis sowie in Tages- und Wochenzeitungen von der Financial Times Deutschland über die Neue Zürcher Zeitung und die VDI nachrichten bis zur Zeit schwerpunktmäßig über politische und rechtliche Themen rund um Internet und Informationstechnik. Buchveröffentlichungen: „Medien, Internet, Krieg: Das Beispiel Kosovo." (2004), „Krieg und Internet: Ausweg aus der Propaganda?" (2004) und „Das Phänomen Berlusconi" (1996).

Christiane Kunath arbeitet seit September 1991 als Systemadministratorin bei der BStU in Berlin. Neben der Administration verschiedener Betriebssysteme (Novell

Netware, Windows 2000, SuSE Linux) gehörte in den letzten Jahren vor allem der Aufbau des Intranets und des Firewallsystems der BStU zu ihrem Aufgabenbereich. Seit Oktober 2002 war sie als stellvertretende Projektleiterin im Referat IT/TK für die konzeptionelle Erarbeitung und Durchführung von Teilschritten der Migration der Server und Clients bei der BStU verantwortlich.

Katja Luther ist Diplombiologin und Studentin der Informatik an der Technischen Universität Berlin mit den Schwerpunkten „Informatik und Gesellschaft" und „Softwaretechnik". Zur Zeit schreibt sie an ihrer Diplomarbeit zum Thema „künstliche Immunsysteme". Sie hat für dieses Buch das Kapitel „Recht und Politik" betreut.

Bernd Lutterbeck (Prof. Dr. iur.) studierte Rechtswissenschaften und Betriebswirtschaftslehre in Kiel und Tübingen. Danach folgten wissenschaftliche Tätigkeiten an den Universitäten Regensburg (1969–1971, Fachbereich Rechtswissenschaft) und Hamburg (1974–1978, Dozent am Fachbereich Informatik) sowie 1976 die Promotion zum Dr. iur. an der Universität Regensburg. Von 1978–1984 war er Beamter beim Bundesbeauftragten für den Datenschutz in Bonn. Seit 1984 ist Bernd Lutterbeck Professor für Wirtschaftsinformatik an der Technischen Universität Berlin mit den Schwerpunkten Informatik und Gesellschaft, Datenschutz- und Informationsrecht, Verwaltungsinformatik. Seit 1995 nimmt er für die Action Jean Monnet der Europäischen Union Brüssel an der Technischen Universität Berlin eine Professur für humanwissenschaftliche Fragen der europäischen Integration wahr. Seine aktuelle Arbeitsschwerpunkte sind E-Government, Theorie und Praxis der *property rights*, Open Source und *European Governance*. Er ist einer der Herausgeber dieses Jahrbuchs.

Christian Maaß (Dipl.-Kfm.) ist wissenschaftlicher Mitarbeiter am Lehrstuhl für Betriebswirtschaftslehre, insbesondere Organisation und Planung, an der FernUniversität Hagen.

Eric Maskin ist A.O.-Hirschman-Professor Sozialwissenschaften am Institute for Advanced Study der Universität Princeton. Vorher unterrichtete er am MIT und der Harvard-Universität. Seine Arbeitsgebiete umfassen verschiedene Bereiche der Ökonomik, einschließlich Spieltheorie, Anreiztheorie und Theorie des gemeinschaftlichen Handelns (*social choice*).

Jason Matusow ist der Direktor der Shared-Source-Initiative von Microsoft. Er ist verantwortlich für die Business Strategy und Implementierung des globalen Programmes zur Lizenzierung von Microsoft Source Code und arbeitet dazu mit Regierungen, Unternehmen und Universitäten zusammen.

Stefan Meretz machte zuerst 1989 seinen Abschluss als Diplom-Ingenieur an der Technischen Universität Berlin. Vier Jahre später absolvierte er erfolgreich das Studium der Informatik (Diplom) und Promotion (Ing.). Er ist derzeit tätig bei „Vereinte Dienstleistungsgewerkschaft" und begleitet dort das Projekt di.ver. Seine Schwerpunkte liegen insbesondere im Projekt Oekonux (http://oekonux.de), Open Theory (http://opentheory.org) und Kritische Informatik (http://kritische-informatik.de) sowie Freie Software.

Stefan Merten studierte Informatik an der Universität Kaiserslautern und machte dort 1992 sein Diplom. Derzeit arbeitet er als technischer Projektleiter bei einer Internetfirma. Neben der Teilnahme an kleineren Freie-Software-Projekten ist er beteiligt am Projekt Oekonux (http://www.oekonux.de) und Open Theory (http://www.opentheory.org).

Jörg Meyer (Dr.) studierte Physik an der Universität Hamburg und promovierte dort bis 1992. Danach begann er seinen beruflichen Werdegang als Systementwickler und Projektleiter bei der has program service GmbH. Seit 1996 arbeitet er bei der Norddeutschen Affinerie AG (http://www.na-ag.com). Nach zwei Jahren als Systemanalytiker wurde er dort Leiter u. a. der Netzwerke und Datenbankentwicklung und seit 2004 auch der gesamten IT-Infrastruktur inklusive der Telekommunikationsanwendungen.

Jan Mühlig (M. A.) ist Vorstand und Gründer der relevantive AG (relevantive.de) und u. a. Mitautor des „Linux Usability Report" von 2003. Er studierte in Regensburg, Mainz, Chicago und Lausanne und hat einen Magister in Soziologie. Sein Interesse gilt insbesondere der projektbegleitenden Usability und damit der Frage, in welchen Prozessen die Benutzerfreundlichkeit für die Open-Source-Entwicklung integriert werden kann.

Jens Mundhenke ist Dipl.-Volkswirt und Dipl.-Kaufmann, studierte Volks- und Betriebswirtschaftslehre an der Universität Passau, dem University College Dublin und der Christian-Albrechts-Universität zu Kiel. Seit 2000 ist er als Wissenschaftlicher Mitarbeiter am Institut für Weltwirtschaft an der Universität Kiel tätig und forscht dort im Bereich der Wettbewerbspolitik in der Neuen Ökonomie sowie der Informations- und Netzwerkgüter. Er promoviert zu Fragen der ökonomischen Bedeutung und der Wirkung von Open Source Software.

Andreas Neumann ist wissenschaftlicher Mitarbeiter am Zentrum für Europäische Integrationsforschung an der Rheinischen Friedrich-Wilhelms-Universität Bonn und Mitglied der Forschungsprojektgruppe „Telekommunikationsrecht". Er gehört zum Redaktionsteam von tkrecht.de, einer WWW-Seite zum deutschen und europäischen Telekommunikations- und Medienrecht, sowie zum Redaktionsteam der Medienrechtsseite artikel5.de.

Denis Petrov (Dipl. Informatiker) studierte an der Russischen Staatlichen Geisteswissenschaftlichen Universität (RGGU Moskau), arbeitet seit 2000 bei der Tembit Software GmbH als Softwareentwickler und Berater im eHealthcare Bereich.

Markus Pizka (Dr.) ist wissenschaftlicher Assistent am Lehrstuhl für Software & Systems Engineering der Technischen Universität München. Im Rahmen seiner Promotion über verteilte Betriebssysteme hat er von 1995–1999 u. a. die OSS-Produkte GNU GCC und Linux adaptiert und eigene OSS-Projekte initiiert. Seit 2001 konzentriert sich seine wissenschaftliche Arbeit auf die Wartung und Evolution großer Softwaresysteme.

Sebastian Redenz lebt in Mannheim und studiert an der dortigen Universität Volkswirtschaftslehre. In seiner Freizeit schreibt er Rezensionen für *De:Bug*. Sebastian Redenz war in früheren Module Groups ab 1998 involviert und veröffentlichte auf einigen Musik. Seit 2001 betreut er mit *Thinner* und *Autoplate* zwei der bekanntesten Netlabel Projekte (http://www.thinnerism.com).

Wolfram Riedel ist Magisterstudent im Hauptstudium an der Technischen Universität Berlin. Seine Hauptfächer sind Kommunikationswissenschaft (Grundlagen von Sprache und Musik) und Informatik (Schwerpunkt Informatik und Gesellschaft). Für dieses Buch hat er das Layout erstellt und das Kapitel „Technik" betreut.

Jan Schallaböck ist zur Zeit Rechtsreferendar am Landgericht Neuruppin. Er war im vergangenen Jahr als Mitarbeiter der Heinrich-Böll-Stiftung mit dem zivilgesellschaftlichen Monitoring des UN-Weltgipfels der Informationsgesellschaft in Genf befaßt. Zuvor war er unter anderem beteiligt an dem Projekt *Bundestux*, das sich für den Einsatz von freier Software in der Bundestagsverwaltung einsetzte, sowie an der Konferenz *Save Privacy*, die die Erosion des von Bürgerrechten zur Zeit der ersten Anti-Terror-Gesetzgebung analysierte.

Ewald Scherm (Univ.-Prof. Dr.) ist Inhaber des Lehrstuhls für Betriebswirtschaftslehre, insbesondere Organisation und Planung, an der FernUniversität Hagen.

Broder Schümann studiert Informatik (Dipl.-Ing.) an der Universität Kiel. Zunächst als Praktikant, dann als freier Mitarbeiter an GENOMatch beteiligt, arbeitet er derzeit im Rahmen seiner Diplomarbeit an einer Erweiterung zu GENOMatch.

Walter Seemayer ist der National Technology Officer der Microsoft Deutschland GmbH. In dieser Funktion ist er für die Unterstützung und Zusammenarbeit mit mit dem Public Sector zuständig. Seine Themenschwerpunkte sind Microsofts Technologie Vision und Strategie, High-Tech Start-Ups, Kosten und Nutzen von IT-Investitionen, Offene Standards, Sicherheit und Datenschutz. Er engagiert sich in verschiedenen Gremien und Lenkungsausschüssen, die sich mit Thema Innovation im IT Bereich beschäftigen wie z. B. dem Münchner Kreis, der Software Offensive Bayern und dem Beirat des Fraunhofer e-Government Zentrums.

Patrick Stewin studiert seit Oktober 2000 Informatik an der Technischen Universität Berlin. Die Schwerpunkte seines Studiums setzt er in den Fachgebieten Intelligente Netze und Management Verteilter Systeme (im Studiengebiet Betriebs- und Kommunikationssysteme) und Informatik und Gesellschaft (am Institut für Wirtschaftsinformatik). Im Rahmen seines Studiums absolviert er im Wintersemester 2004/05 ein Praktikum bei der Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung im Bundesministerium des Innern und arbeitet dort am Open-Source-Software-Kompetenzzentrum mit. Er wirkte bereits am „Open Source Jahrbuch 2004" mit und hat in diesem Buch des Kapitel „Fallbeispiele" betreut.

Joachim Sturm (Dr.) ist seit April 2004 Leiter der KBSt (Referat IT 2 im IT-Stab des BMI). Aufgabe der KBSt ist primär die Beratung der obersten Bundesbehörden beim Einsatz von IuK. Sie gibt dazu Empfehlungen heraus und koordiniert wesentliche Entwicklungsvorhaben. Sie ist darüber hinaus auch für die Regierungskommunikation (IVBB) zuständig . Des weiteren zeichnet sie sich verantwortlich für Standadisierungsfragen (SAGA) und den Einsatz von OSS in der Bundesverwaltung (Migrationslitfaden). Als Leiter der KBSt ist Herr Sturm Vorsitzender des IMKA und Vertreter des Bundes im KoopA ADV.

Mark Tins (Dipl.-Kaufm.) studierte Betriebswirtschaftslehre an der Ludwig-Maximilians-Universität München. Seine Schwerpunkte waren Innovationsmanagement am Lehrstuhl von Prof. Dietmar Harhoff und Informations- und Kommunikationsmanagement am Lehrstuhl von Prof. Arnold Picot. Im Rahmen seiner Ausbildung verbrachte er jeweils einen mehrmonatigen Studienaufenthalt in Ohio, USA und an der Katholieke Universiteit in Leuven, Belgien. Im Februar 2004 schloss er sein Studium mit der empirischen Diplomarbeit zum Thema „Informelle Entwicklungskooperationen in der Embedded Software Branche" ab. Seit Anfang 2004 ist er im Produktmanagement in der Mobilfunkbranche tätig.

Siva Vaidhyanathan (http://sivacracy.net) ist der Autor von den Büchern „Copyrights and Copywrongs: The Rise of Intellectual Property and How it Threatens Creativity" (New York University Press, 2001) und „The Anarchist in the Library: How the Clash Between Freedom and Control is Hacking the Real World and Crashing the System" (Basic Books, 2004). Er hat bereits eine Vielzahl an Beiträgen für Zeitschriften wie *The Chronicle of Higher Education*, *The New York Times Magazine*, *MSNBC.COM*, *Salon.com*, *openDemocracy.net* und *The Nation* geschrieben. Nach fünfjähriger Tätigkeit als professioneller Journalist hat Vaidhyanathan an der University of Texas in Amerika Studien promoviert. Er hat bisher an der Wesleyan University und an der University of Wisconsin gelehrt und ist derzeitig Dozent fuer Kultur- und Kommunikationswissenschaften an der New York University.

Tile von Damm leitet die unabhängige Analyseorganisation Perspektiven Globaler Politik (*PerGlobal*). Er arbeitet und forscht zur Ausgestaltung der Europäischen Union, zu internationalen und europäischen Umwelt- und Nachhaltigkeitsprozessen, zur Stärkung einer euopäischen Zivilgesellschaft, zu Partizipationsprozessen sowie zur Informationsgesellschaft. Als NGO-Vertreter war er auf den UN-Gipfeln zur Nachhaltigen Entwicklung und zur Weltinformationsgesellschaft. Er studierte Politikwissenschaften in Berlin und Marburg und war 1996-1999 bei *Sony Music Germany* (http://www.perglobal.org, http://www.nachhaltiges-europa.de).

Eric von Hippel leitet die Forschungsgruppe für Technological Innovation and Entrepreneurship an der MIT Sloan School of Management. Seine Forschungsschwerpunkte sind die Quellen und Ökonomie von Innovation.

Jakob Voß studiert Informatik und Bibliothekswissenschaft an der Humboldt–Universität zu Berlin. Er ist Mitglied des Vorstands von *Wikimedia Deutschland e. V.*

Sören Wurch ist Student der Informatik an der Technischen Universität Berlin. Zur Zeit stellt er seine Diplomarbeit zum Thema „elektronisches Publizieren“ fertig.

Sebastian Ziebell studiert seit Oktober 2000 Informatik an der Technischen Universität Berlin. Die Schwerpunkte in seinem Studium sind „Softwaretechnik“ und „Informatik und Gesellschaft“. Er hat das Kapitel „Gesellschaft“ betreut.

Bastian Zimmermann ist Student an der Technischen Universität Berlin. Die Schwerpunkte in seinem Studium sind „Softwaretechnik“ und „Informatik und Gesellschaft“. Er hat das Qualitätsmanagement und die Übersetzung verschiedener Artikel im Buch übernommen.

Auch zu empfehlen – Das *Open Source Jahrbuch 2004*
Erhältlich unter http://www.opensourcejahrbuch.de