Freie Software

Geschichte, Dynamiken und gesellschaftliche Bezüge

Volker Grassmuck
vgrass@rz.hu-berlin.de

Ver 1.0
September 2000

Korrekturen, Ergänzungen, Anregungen willkommen.

Die vorliegende Ausarbeitung ist im Rahmen des Forschungsprojektes “Von der Ordnung des Wissens zur Wissenordnung digitaler Medien” entstanden. Sie beruht unter anderem auf dem **Fachgespräch Open Source-Software des BMWi** am 15.7.1999 und der Konferenz **Wizards of OS. Offene Quellen und Freie Software** am 16.-17.7.1999, beide im Haus der Kulturen der Welt in Berlin. Die Arbeit erhielt finanzielle Unterstützung von der DFG und dem BMWi.

online unter
http://mikro.org/Events/OS/text/freie-sw.html und
http://mikro.org/Events/OS/text/freie-sw.pdf (dieser Text)

Einleitung

Daß Menschen in freier Kooperation ohne primäres Interesse am Gelderwerb, ohne sich je getroffen zu haben und ohne eine Leitung, die ihnen sagt, was sie zu tun haben, hochwertige Software schreiben, ist eine erstaunliche Tatsache. Zumal in einer Zeit, da Wissen in einem Ausmaß und mit einer Reichweite kommodifiziert wird, wie es nie zuvor auch nur denkbar war. Im Sommer 1998 trat diese verblüffende Entwicklung ins Licht der Öffentlichkeit. Software war mit dem Aufkommen des PC-Marktes zu Beginn der achtziger Jahre zu einem regulären Massenprodukt geworden. Firmen wie Microsoft, Corel oder IBM boten Betriebssysteme, Anwendungen und Netze als Standardware im Regal an. Verblüffend war nun, daß in einem über zwanzig Jahre gereiften Marksegment plötzlich Konkurrenz aus einer unwahrscheinlichen Richtung auftauchte. Erfolgs-Software wie Linux, Perl oder XFree86 wird von Gruppen locker zusammenarbeitender vermeintlicher Hobbyisten erstellt, die meist nicht einmal eine ordentliche Rechtsform hatten. Und sie verkauften keine Produkte, sondern verschenkten eine Teilhabe an ihrem gemeinsamen Software-Entwicklungsprozeß. Während die Industrie den Schutz ihres exklusiven *geistigen Eigentums* immer weiter ausbaut (die Entscheidung des US-Verfassungsgerichts von 1981, die erstmals die grundsätzliche Patentierbarkeit von Software bekräftige, war ein wichtiger Schritt in diese Richtung), liegt die Stärke des alternativen Modells darin, den *geistigen Reichtum* frei mit aller Welt zu teilen und gemeinsam fortzuschreiben.

Im Zentrum der Aufmerksamkeit stand das Unix-artige PC-Betriebssystem GNU/Linux. Philosophen und Evangelisten gab es auch früher, doch ihr ‘Manifest’ erhielt die Open-Source-Bewegung Anfang 1998 mit Eric Raymonds Aufsatz “The Cathedral and the Bazaar”.[1] Darin beschreibt er die Faustregeln für eine verteilte, offene, locker gekoppelte Zusammenarbeit am selben komplexen Software-Projekt durch Tausende von über die ganze Welt verstreuten Menschen. Sein Paradebeispiel für den Erfolg einer solchen unwahrscheinlichen Organistationsform ist das von Linus Torvalds initiierte Betriebssystemkern-Projekt Linux. “In fact, I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model.” Open Source stellt die konventionelle Logik auf den Kopf: verteilte Gruppen von Gleichen sind effizienter als Chef-geleitete, hierarchische Systeme. Wer die Früchte seiner Arbeit verschenkt, hat mehr davon. Eigennutz und Anerkennung motivieren effektiver als Geld.

Fragt man Eingeweihte, wie denn solche freien Projekte zustande kommen und welche Ingredienzen es dazu braucht, erfährt man, daß das ganz einfach sei: “In dem Moment, wo man Leute mit einer guten Ausbildung und viel Freizeit zusammentut mit einer guten Kommunikationsinfrastruktur, also gut verfügbarer Netzbandbreite, sorgen die von selbst dafür, daß sie sich nicht langweilen und fangen irgendwelche seltsame Projekte an, wie zB. das KDE-Projekt.[2] Das ist die perfekte Umgebung, wie sie z.B. an der Uni mit ausreichend vielen Assistenten herrscht oder in anderen Umgebungen, wo man diese Faktoren vernünftig zueinander bringen kann.”[3]

Zu den spektakulären Entwicklungen des Jahres 1998 gehören Netscapes Freigabe des Quellcodes für seinen Browser und der Deal zwischen IBM und der Apache-Gruppe, um

---

[1] Raymond 1998

[2] das K-Desktop-Environment, eine graphische Benutzeroberfläche für Linux, s.u.

[3] K. Köhntopp auf Fachgespräch 1999

deren http-Server in IBMs WebSphere-Suit zu integrieren. Im Mai kündigte Corel die Portierung von WordPerfect auf Linux an. Im Juli Oracle und Informix mit ihren Datenbanken. Netscape, Cygnus, Sendmail, McAfree, Caldera, RedHat, Cisco bieten quelloffene Software als Produkt an. Sun Microssystems, IBM, Novell, Hewlett-Packard, SAP, Orcale, Dell, Compaq, Silicon Graphics begannen ebenfalls, quelloffene Software zu verkaufen und zu unterstüzen. IBM schloß Lizenzabkommen mit RedHat, SuSE, Caldera und Pacific HiTeck über Marketing, Schulung, Support und Entwicklung für Linux. Auch Computer Associates, das drittgrößte Softwarehaus der Welt, kündigte seine Unterstüzung für Linux an. Im Dezember baut das CLOWN Projekt, live in der WDR Computer-Nacht mitzuverfolgen, einen Linux-Cluster mit 550 Knoten, rekordträchtig nach allen Maßstäben.

Die Wellen schlugen so hoch, daß selbst Microsoft (MS) eine hausinterne Expertise über Open Source in Auftrag gab. Durch die Enthüllung dieses vetraulichen, sog. "Halloween-Dokuments"[4] im November wurde sichtbar, daß MS Open-Source-Software als direkte und kurzfristige Bedrohung seines Profits und seiner Plattform ansieht. Mehr noch sieht der Autor im freien Austausch von Ideen einen Vorteil, der von MS nicht repliziert werden könne, und deshalb eine langfristige Bedrohung des *Mindshare* des Unternehmens bei den Software-Entwicklern darstelle. Zur Abwehr der Gefahr empfiehlt er u.a. eine Strategie, die MS bereits gegen Java einsetzte: MS erweitert offene Standards um proprietäre Funktionen, die durch Marktsättigung als "de facto"-Standard durchgesetzt werden. Die Befürchtungen waren nicht unbegründet. Tatsächlich hat sich GNU/Linux als formidabler Konkurrent um den Server-Markt gegen Windows-NT etabliert. Im Oktober, im Vorfeld des Kartellverfahrens, hatte Microsoft Linux als Beleg dafür angeführt, daß es kein Monopol im Betriebssystemmarkt habe.

Als selbst Forbes Magazine in seiner Ausgabe vom August 1998, mit Linus Torvalds auf dem Titel, das Loblied von Open Source und Hackern sang, mußte man sich fragen, ob dieses Phänomen tatsächlich an die Grundfesten des Kapitalismus rührte. Im Oktober kündigen Intel und Netscape Investitionen in Red Hat Software an. Zu Hollywood-Ruhm gelangte Linux durch den Film Titanic, dessen Computergraphiken auf einem Linux-Cluster gerechnet wurden. Sogar in der Kunstwelt fand es Anerkennung, als Linus Torvalds stellvertretend für sein Projekt der Linzer Prix Ars Electronica 1999 verliehen wurde. Die Zahl der installierten Linux-Systeme stieg auf einige zehn Millionen weltweit. Der Markt verzeichnete eine regelrechte Linux-Hysterie. Auf der CEBIT 1999 konnte man sich vor lauter Pinguinen -- dem "Tux" genannten Maskottchen von Linux -- kaum retten. Jedes Software-Unternehmen mit einer Warenwirtschaft, einer Zahnärtzelösung oder einem eCommerce-System kündigte an, dieses Produkt jetzt auch für Linux anzubieten.[5]

Wer sich erinnern konnte, wußten schon damals, daß freie Software nichts Neues war. Tatsächlich lassen sich verschiedene Phasen dieser Bewegung unterscheiden. Vor 1980 macht es wenig Sinn von 'freier' Software zu sprechen, da es keine 'unfreie' gab. Der Computer-Markt beruhte allein auf Hardware, zu der Software als eine nicht schützensfähige Zugabe galt. Auch die Software, aus der das Internet besteht, entstand vorwiegend in einer akademischen Umgebung ganz selbstverständlich in einem offenen kooperativen Prozeß, ohne Barrieren wie Copyrights und Patente. Die 80er Jahre stehen dann unter dem Zeichen der proprietären Schließung von Software durch Microsoft (MS-DOS) und AT&T (Unix)

---

[4] Valloppillil 1998

[5] für eine Linux-Chronologie des Jahres 1998 s. Linux Weekly News, Januar 1999, http://lwn.net/1999/features/1998timeline/

und der Gegenbewegung des GNU-Projekts,[6] Quelle und Sammelbecken für viele wichtige freie Projekte dieser Zeit. Fairerweise muß man dazu sagen, daß auch auf den geschlossenen Plattformen PC/M, MS-DOS und MS-Windows eine Fülle von Free- und Shareware entstanden ist. Windows-Archive haben durchaus auch einen Basar-Charakter. In den 90er Jahren kommt der Wechsel von Mainframe und Workstation zum PC voll zum Tragen, und die 'Vollbesiedlung' des Internet beginnt. Eingeläutet wird die Dekade von Linus Torvalds' Projekt, Unix auf den PC zu übertragen. Auch dem Berkeley-Unix wird durch seine Portierung auf die Intel-Architektur neues Leben eingehaucht. Mit der Erschließung der billigen und massenhaft zugänglichen Plattform durch die beiden Unix-Kerne und durch Basiswerkzeuge wie dem GNU-C-Compiler wurde der Software-Pool des GNU-Projekts nutzbar. Weitere Projekte, wie XFree86 und der Apache-Webserver, wandert aus der Welt der großen Rechner herüber und neue, wie das Bildverarbeitungsprogramm GIMP, entstanden. Da die freien Unixe sich an den Posix-Standard[7] halten, steht ihnen darüberhinaus -- zumindest auf Quellcodeebene -- das gesamte Software-Angebot unter Unix zur Verfügung. Betriebssysteme, Werkzeuge und Applikationen erzeugten eine sich gegenseitig verstärkende Dynamik, die schließlich 1998 so spektakulär eine breitere Öffentlichkeit erreichte.

Erst im Zuge dieser öffentlichkeitswirksamen Entwicklungen beginnt das Management in den Unternehmen, freie Software überhaupt als brauchbare Alternative im produktiven Einsatz anzuerkennen. Diese Akzeptanz ging graswurzelartig von den Technikern aus. Noch 1998 war eine häufige Konstellation, daß Firmensprecher öffentlich abstritten (und u.U. auch gar nicht wußten), daß in ihrem Unternehmen freie Software eingesetzt wird, während die Systemadministratoren aus praktischen oder Kostengründen ihre Server mit Linux und Apache betrieben. Was vorher als 'unseriös' galt, wurde mit der breiten Aufmerksamkeit für freie Software zu etwas, mit dem ein Unternehmen einen Anerkennungsbonus erlangen konnte. Firmen wie Sixt oder Ikea konnten ihre hausinternen Linux-Systeme jetzt gewinnbringend in ihrem Marketing präsentieren. Nirgends sonst wird der Übergang von der Waren- zur Dienstleistungsgesellschaft, von Produkt zu Prozeß, so deutlich sichtbar, wie bei der Software.

Auch in der Verwaltung wird freie Software inzwischen zur Kenntnis genommen. So evaluierte z.B. eine Arbeitsgruppe des Berliner Senats, die 1999 eine Empfehlung zur Verwendung von Betriebssystemen in der gesamten Berliner Bundesstruktur erarbeiten sollte, GNU/Linux gegenüber Windows NT.[8] Noch weiter gingen die beiden französischen Abgeordneten Pierre Laffitte und René Trégouët mit ihrer Gesetzesvorlage, derzufolge Frankreichs Regierung und Verwaltung ausschließlich Software verwenden soll, die frei von geistigen Eigentumsrechten und deren Quellcode verfügbar ist, da nur quelloffene Systeme die erforderliche Evaluierung zulassen. Die beiden Abgeordneten sehen das Internet als den hauptsächlichen Weg, auf dem Staat und Bürger zukünftig kommunizieren werden, daher dürfe sich die Verwaltung nicht vom Wohlwollen von Software-Verlegern abhängig machen.[9] Unabhängigkeit, Sicherheit und die vergleichsweise neue benutzerfreundliche Einsetzbarkeit von freier Software im Büroalltag hebt auch die deutsche Koordinierungs- und

---

[6] das von Richard Stallman initiierte freie Unix-Derivat, s.u.

[7] Portable Operating System Interface for Computer Environments, IEEE Standard 1003.1-1988

[8] Ein einflußreicher, akribischer Vergleich der beiden ist Kirch 1998

[9] Lettice 1999; die beiden Gesetzentwürfe: http://www.senat.fr/grp/rdse/page/forum/texteloi.html; eine öffentliche Debatte dazu im Forum: http://www.senat.fr/Vforum/5/forum.html

Beratungsstelle der Bundesregierung für Informationstechnik in ihrer aktuellen Analyse hervor und empfiehlt eine “Migration zu Open-Sources-Software”.[10]

Das Basar-Verfahren der quelloffenen Software-Entwicklung beruht auf kollektiver Intelligenz, Peer-Reviews und einer breiten funktionalen Bewertung der Systeme. Die Voraussetzung dafür ist die ungehinderte Weitergabe und Weiterverwendung des Quelltextes, der Datenformate und der Dokumentation der betreffenden Software. Freie Software trägt nicht nur dem Prozeßcharakter der digitalen Ware Rechnung und bringt stabilere, interoperable, portierbare, weiterverwendbare Software hervor, sondern speist auch den Pool des Gemeingutes: Wissen, das allen gehört.

Dieses Kapitel exemplifiziert die Perspektive, die im vorangegangenen mit der Wissens-Allmende eröffnet wurde. Durch die große gesellschaftliche Bedeutung, die die freie Software erlangt hat, scheint es uns geboten, ausführlich auf das Phänomen einzugehen. Die Kooperationsmechanismen, die technischen, sozialen, kulturellen und rechtlichen Kontexte und die Auswirkungen von freier Software werden dargestellt. Anhand einzelner Projekte werden ihr Entstehen, ihre Organisations- und Kommunikationsformen und die Motive ihrer Beteiligten erläutert. Es folgt eine Diskussion der gesellschaftlichen und wirtschaftlichen Potentiale der freien Software. Ihre Schnittstelle zur konventionellen Wirtschaft sind die verschiedenen Lizenzen. Firmen, die Dienstleistungen rund um freie Software anbieten oder sie in ihren Unternehmensprozessen einsetzen, werden vorgestellt. Es schließt sich das für jede Software wichtige Thema der Sicherheit an, und am Schluß erfolgt ein Blick auf die weiteren Entwicklungen und die Implikationen für die digitale Wissensordnung.

[10] KBSt-Brief Nr. 2/2000

Geschichte

“Wissenskommunismus” der Wissenschaften

Seit den Athenern gehört es zur Universität, daß das von ihr erzeugte und durch sie weitergegebene Wissen, anders als das in geschlossenen und gar geheimen Forschungsstellen der Staaten oder der Industrien üblich ist, ohne den Schutz von Patenten und Copyright zirkulieren können muß. Die im Hochmittelalter entstandenen europäischen Universitäten bildeten einen Medienverbund aus Verarbeitung des gesprochenen Wortes zu handschriftlichen Büchern, Speicherung in Bibliotheken und Übertragung von Texten in einem eigenen Universitätspostsystem. In der frühen Neuzeit übernahmen dank Gutenbergs Erfindung Verlage die Produktion von Büchern, und die entstehenden Territorialstaaten und später Nationalstaaten beanspruchten das Postmonopol. Die Informationsverarbeitung, so Friedrich Kittler, wurde einer Hardware übertragen, die in den geschlossenen Kreisen der militärischen Nachrichtentechniken entstand. Die Computer-Software dagegen sei eine Schöpfung der Universität gewesen. Die universale Turing-Maschine stammte als Konzept und als Software aus einer akademischen Dissertation. “Ganz entsprechend stammt die noch immer herrschende von-Neumann-Architektur von einem, der es vom Göttinger mathematischen Privatdozenten schließlich zum Chefberater des Pentagon brachte. Auf diesem Weg zur Macht hat das Wissen, das in Computer und ihre Algorithmen versenkt ist, einmal mehr jene Schließung erfahren, die einst bei der Übernahme der Universitäten durch die Territorialstaaten drohte.”[11]

Solcher realen Vereinnahmungen zum Trotz enwirft die Gelehrtenrepublik des 19. Jahrhunderts eine akademische Wissenschaftsverfassung, die auf der Freiheit von Lehre und Forschung beruht. Konstitutiv für diese klassische Wissensordnung Humboldtscher Prägung und fortgeschrieben in der Forschungsgemeinschaft des 20. durch Autoren wie Weber, Popper, Merton, Spinner usw. sind vier große Abkopplungen:

- Die Trennung von Erkenntnis und Eigentum: Forschungsergebnisse müssen veröffentlicht werden, um sie in einem *Peer Review*-Prozeß überprüfen, replizieren, kritisieren und fortschreiben zu können. Das ist es, was Robert Merton mit dem "Wissenskommunismus" der Wissenschaften meinte.[12]
- die Trennung von Ideen und Interessen
- die Trennung von Theorie und Praxis
- die Trennung von Wissenschaft und Staat: Lehre und Forschung folgen keinen externen Anweisungen. D.h. nicht, daß sie nicht öffentlich finanziert werden dürften, ganz im Gegenteil. Tatsächlich wurde die Grundlagenforschung für die neue Ordnung digitaler Medien, also der Computer und Datennetze, mit öffentlichen Mitteln betrieben.[13]

---

[11] Friedrich Kittler, Wizards 7/1999

[12] Robert K. Merton, The Sociology of Science, Chicago & London, Chicago Univ. Press, 1973: S. 273 ff., zitiert nach Spinner 1998: 36

[13] Spinner 1994: 15 f.

Der für die freie Software wesentliche Punkt ist die "Abkopplung der Ideenwirtschaft von der normalen Güterwirtschaft."[14] Mit seiner Veröffentlichung wird das Wissen zum Gemeingut der Forschungsgemeinschaft. Es kann von Kollegen frei nachvollzogen, überprüft und weiterentwickelt werden und in der Lehre frei der Reproduktion der Wissensträger in der nächsten Generation dienen. Durch diese fruchtbaren Bedingungen im 'Sondermilieu' der Wissenschaften können die parallelen, kollektiven Bemühungen Ergebnisse hervorbringen, die kein Einzelner und kein einzelnes Team produzieren könnten. Die einzelne Wissenschaftlerin erhält im Wissenskommunismus als Anerkennung für die von ihr erarbeiteten Erkenntnisse keine Geldzahlungen -- um von dieser Notwendigkeit freigestellt zu sein, allimentiert sie der Staat --, sondern ein symbolisches Entgeld in Form von fachlicher Reputation, wie sie sich z.B. an der Zahl der Einträge im *Citation Index* ablesen läßt. Statt einem Monopolverwertungsrecht, wie es das Patentsystem für Erfindungen von industriellem Wert gewährt, steht hier das Recht auf Namensnennung im Vordergrund.

Die Wissensordnung dieses Sondermilieus strahlt über ihren eigentlichen Geltungsbereich hinaus auf seine Umwelt in der modernen, demokratischen Gesellschaft aus, mit der zusammen sie entstanden ist.

> "Der Wissenstransfer in das gesellschaftliche Umfeld konnte unter günstigen Bedingungen (Rechtsstaat, Demokratie, liberale Öffentlichkeit) wesentliche Bestandteile dieser Wissensordnung in die 'Wissensverfassung' der Gesellschaft einfließen lassen. Die freie wissenschaftliche Forschung, Lehre, Veröffentlichung findet so ihre Ergänzung in der 'Freien Meinung' des Bürgers[15] und verwandter Wissensfreiheiten, wie in unserem Grundgesetz verankert. So spannt sich der Bogen der ordnungspolitischen Leitvorstellungen, mit Abstrichen auch der positiven Regulierungen und praktischen Realisierungen, vom Wissenskommunismus der Forschungsgemeinschaft bis zur informationellen Grundversorgung in der Informationsgesellschaft und dem geforderten weltweiten freien Informationsfluß..."[16]

Internet

Das Internet ist mediengeschichtlich eine Anomalie. Übliche Modelle der Medien- wie allgemein der Technikgenese laufen vom Labor über die Entwicklung hin zur Anwendungsreife bis zur gesellschaftlichen Implementierung entweder als staatliche Militär- oder Verwaltungskommunikation, als wirtschaftliches Kontroll- und Steuerungsinstrument oder als Massenprodukt der Individualkommunikation oder Massenmedien. Anders hingegen im Falle von akademischen Datennetzen. Hier gab es in den ersten Jahren keine Trennung zwischen Erfindern, Entwicklern und Anwendern.

Die Informatik hat im Netz nicht nur ihren Forschungsgegenstand, sondern zugleich ihr Kommunikations- und Publikationsmedium. Es ist gleichzeitig Infrastruktur und Entwicklungsumgebung, die von innen heraus ausgebaut wird. Innovationen werden von den

---

[14] Spinner 1994: 91

[15] an anderer Stelle (Spinner 1998: 66) fügt Spinner die öffentlich-rechtlichen Medienlandschaft hinzu.

[16] Spinner 1998: 48 f.

Entwickler-Anwendern in der Betaversion (d.h. ohne Garantie und auf eigene Gefahr) in die Runde geworfen, von den Kollegen getestet und weiterentwickelt. Darüber hinaus stellt sie den anderen, zunehmend computerisierten Wissenschaften die gleiche Infrastruktur zur Verfügung. Der Zugang zu Rechenressourcen, der Austausch innerhalb einer weltweiten *Community* von Fachkollegen, das Zur-Diskussion-Stellen von *Pre-Prints*, die Veröffentlichung von Konferenzreferaten und Datenbanken im Internet -- all dies gehört seit den achtziger Jahren zu den täglichen Praktiken in der Physik und Astronomie, der Informatik selbst und zunehmend auch in den 'weicheren' Wissenschaften. Schließlich ist das Weiterreichen der Grundwerkzeuge an die Studenten Teil der wissenschaftlichen Lehre. Da das Netz, anders als die meisten Laborgeräte, keinen eng definierten Anwendungsbereich hat, sondern eben Medium ist, kommen hier auch studentische private und Freizeitkulturen auf -- eine brisante Mischung aus Hi-Tech und Hobbyismus, *Science* und *Science Fiction*, Hackern und Hippies.

Die Geschichte des Internet läßt sich grob in drei Phasen einteilen: in der Frühphase ab Mitte der sechziger Jahre werden die Grundlagen gelegt, die Technologie demonstriert und zur Anwendungsfähigkeit entwickelt. Zeitgleich mit dem Wechsel von der militärischen zur akademischen Forschungsförderung Ende der Siebziger beginnt das Wachstum und die internationale Ausbreitung des Internet. In dieser Zeit gedieh das, was gemeinhin mit der 'wilden Phase' des ursprünglichen Internets assoziiert wird: eine 'Gabentausch'-Ökonomie für Software und Information, eine Graswurzel-basierte Selbstorganisation, emergierende *Communities* und der Hacker-Geist, der jede Schließung, jede Beschränkung des Zugangs und des freien Informationsflusses zu umgehen weiß. 1990 wird das ARPAnet abgeschaltet, und es beginnt die kommerzielle Phase des Internet.

Frühphase

In den späten fünfziger Jahren leitete J.C.R. Licklider eine Forschungsgruppe beim US-Rüstungslieferanten Bolt, Beranek and Newman (BBN), die auf einer PDP-1 eines der ersten *Time-Sharing*-Systeme bauten. Computerhersteller und die meisten Vertreter des Informatik-Establishments waren der Ansicht, daß *Time-Sharing* eine ineffiziente Verwendung von Computer-Ressourcen darstelle und nicht weiter verfolgt werden solle. Lickliders Argument war umgekehrt, daß Rechner für eine Echtzeit-Interaktion (für "kooperatives Denken mit einem Menschen") zu schnell und zu kostspielig seien, weshalb sie ihre Zeit zwischen vielen Nutzern aufteilen müssten. Licklider war auch der Architekt des MIT Projekts MAC (*Multiple-Access Computer* oder *Machine-Aided Cognition* oder *Man And Computer*). 1962 wechselte er von BBN zur *Advanced Research Projects Agency* (ARPA) des US-Verteidigungsministeriums, wo er Leiter des *Command and Control Research* wurde, das er in *Information Processing Techniques Office* (IPTO) umbenannte.[17]

Seine Erfahrungen mit *Time-Sharing*-Systemen erlaubten es ihm, eine Neudefinition vom Computer als Rechenmaschine zum Computer als Kommunikationsgerät vorzunehmen. Als Leiter des ARPA-Forschungsbereiches war er nun in die Lage versetzt, diesen Paradigmenwechsel in der Netzplanung zur Wirkung zu bringen.

[17] Zu Licklider s. David S. Bennahums Webseite http://www.memex.org/licklider.html, wo sich auch Licks Texte "Man-Computer Symbiosis" (1960) und "The Computer as a Communications Device" (1968, zusammen mit Robert Taylor) finden.

> "The ARPA theme is that the promise offered by the computer as a communication medium between people, dwarfs into relative insignificance the historical beginnings of the computer as an arithmetic engine...
>
> Lick was among the first to perceive the spirit of community created among the users of the first time-sharing systems... In pointing out the community phenomena created, in part, by the sharing of resources in one timesharing system, Lick made it easy to think about interconnecting the communities, the interconnection of interactive, on-line communities of people, ..."[18]

Zeitgleich findet ein telekommunikationstechnische Paradigmenwechsel von leitungsorientierten zu paketvermittelten Konzepten statt. Er geht auf parallele Arbeiten von Paul Baran an der RAND Corporation[19] und von Donald Watts Davies am National Physical Laboratory in Middlesex, England zurück. Die Zerlegung von Kommunikationen in kleine Datenpakete, die, mit Ziel- und Absenderadresse versehen, 'autonom' ihren Weg duch das Netzwerk finden, war Voraussetzung für die verteilte, dezentrale Architektur des Internet. Sie war auch der Punkt, an dem die Geister der Computer- und der Telekommunikationswelt sich schieden.

Die Telefonbetreiber der Zeit waren durchaus an Datenkommunikation und, nachdem nachgewiesen war, daß Paketvermittlung nicht nur überhaupt möglich war, sondern die vorhandene Bandbreite viel wirtschaftlicher nutzt als Leitungsvermittlung, auch an dieser Technik interessiert, doch ihre vorrangigen Designkriterien waren flächendeckende Netzsicherheit, Dienstequalität und Abrechenbarkeit. Diese sahen sie nur durch ein zentral gesteuertes Netz mit dedizierter Leitungsnutzung für jede einzelne Kommunikation gewährleistet. Die Telcos vor allem in England, Italien, Deutschland und Japan unterlegten daher den unberechenbaren Paketflüssen eine 'virtuelle Kanalstruktur'. Auch in diesem System werden Pakete verschiedener Verbindungen auf derselben physikalischen Leitung ineinandergefädelt, aber nur bis zu einer Obergrenze, bis zu der die Kapazität für jede einzelne Verbindung gewährleistet werden kann. Außerdem ist dieses Netz nicht verteilt, sondern über zentrale Vermittlungsstellen geschaltet. Die Spezifikationen dieses Dienstes wurden im Rahmen der Internationalen Telekommunikations-Union verhandelt und 1976 unter der Bezeichnung X.25 standardisiert. Die Bundespost bot ihn unter dem Namen Datex-P an. Damit ist der Gegensatz aufgespannt zwischen einem rhizomatischen Netz, das aus einem militärischen Kalkül heraus von einzelnen Knoten dezentral wuchert, und einer hierarchischen, baumförmigen Struktur, die zentral geplant und verwaltet wird.

Doch zurück zum Internet. Die ARPA-Forschungsabteilung unter Licklider schrieb die verschiedenen Bestandteile des neuen Netzes aus. Das Stanford Research Institute (SRI) erhielt den Auftrag, die Spezifikationen für das neue Netz zu schreiben. Im Dezember 1968 legte das SRI den Bericht "A Study of Computer Network Design Parameters" vor. Zur selben Zeit arbeitete Doug Engelbart und seine Gruppe am SRI bereits an computer-gestützten Techniken zur Förderung von menschlicher Interaktion. Daher wurde entschieden, daß das SRI der geeignete Ort sei, ein *Network Information Center* (NIC) für das ARPAnet einzurichten. Die DARPA-Ausschreibung für ein *Network Measurement Center* ging an die University of California in Los Angeles (UCLA), wo Leonard Kleinrock arbeitete, der seine

---

[18] ARPA draft, III-24 und III-21, zitiert Hauben/Hauben 1996, Chapter 6

[19] s. http://www.rand.org/publications/RM/baran.list.html, besonders "On Distributed Communications Networks" (1964)

Doktorarbeit über Warteschlangentheorie geschrieben hatte. Ebenfalls im UCLA-Team arbeiteten damals Vinton G. Cerf, Jon Postel und Steve Crocker.

Den Zuschlag für die Entwicklung der Paketvermittlungstechnologien, genauer eines *Interface Message Processors* (IMP), erhielt BBN. Dort arbeitete u.a. Robert Kahn, der vom MIT gekommen war und auf den ein Großteil der Architektur des Internet zurückgeht. Die IMPs (Vorläufer der heutigen *Router*) hatten die Aufgabe, die niedrigste Verbindungsschicht zwischen den über Telephonleitungen vernetzten Rechnern (*Hosts*) herzustellen. Die ersten IMPs wurden im Mai 1969 ausgeliefert.

Der Startschuß zum Internet fiel im Herbst 1969, als die ersten vier Großrechner in der UCLA, im SRI, der University of Californiy in Santa Barbara (UCSB) und der University of Utah miteinader verbunden wurden.[20]

Bereits ein halbes Jahr vorher war das erste von Tausenden von *Request for Comments*-Dokumenten (RFC)[21] erschienen, die die technischen Standards des Internet spezifizieren. Diese Standards werden nicht im Duktus eines Gesetzes erlassen, sondern als freundliche Bitte um Kommentierung. Steve Crocker, Autor des ersten RFC, begründete diese Form damit, daß die Beteiligten nur Doktoranden ohne jede Autorität waren. Sie mußten daher einen Weg finden, ihre Arbeit zu dokumentieren, ohne daß es schien, als wollten sie irgendjemandem etwas aufoktroyieren, eine Form, die offen war für Kommentare. RFCs können von jedem erstellt werden. Sie sind als Diskussionspapiere gedacht, mit dem erklärten Ziel, die Autorität des Geschriebenen zu brechen.[22] Meist technische Texte wird in ihnen auch die Philosophie (z.B. RFC 1718) und Geschichte (RFC 2235) des Netzes und seiner Kultur aufgezeichnet und zuweilen sogar gedichtet (RFC 1121). Die freie Verfügbarkeit der Spezifikationen und der dazugehörigen Referenzimplementationen waren ein Schlüsselfaktor bei der Entwicklung des Internet. Aus dem ersten RFC ging ein Jahr später das *Network Control Protocol* (NCP) hervor, ein Satz von Programmen für die Host-Host-Verbindung, das erste ARPANET-Protokoll.

1971 bestand das Netz aus 14 Knoten und wuchs um einen pro Monat,[23] darunter Rechner der verschiedensten Hersteller (DEC-10s, PDP8s, PDP-11s, IBM 360s, Multics, Honeywell usw.). Nach Fertigstellung des NCP und Implementiationen für die verschiedenen Architekturen entstanden jetzt die höheren Dienste Telnet (RFC 318) und FTP (File Transfer Protocol, RFC 454). Ray Tomlinson (BBN) modifizierte ein eMail-Programm für das

---

[20] ein Diagramm dieses ersten 4-Knoten-ARPAnets s. http://www.computerhistory.org/exhibits/internet_history/full_size_images/1969_4-node_map.gif

[21] Erster RFC: “Host Software” von Stephen D. Crocker über die Kommunikation zwischen IMP und dem bereits vorhandenen Host-Rechner; ftp://ftp.denic.de/pub/rfc/rfc1.txt

[22] “The content of a NWG [Network Working Group] note may be any thought, suggestion, etc. related to the HOST software or other aspect of the network. Notes are encouraged to be timely rather than polished. Philosophical positions without examples or other specifics, specific suggestions or implementation techniques without introductory or background explication, and explicit questions without any attempted answers are all acceptable. The minimum length for a NWG note is one sentence. These standards (or lack of them) are stated explicitly for two reasons. First, there is a tendency to view a written statement as ipso facto authoritative, and we hope to promote the exchange and discussion of considerably less than authoritative ideas. Second, there is a natural hesitancy to publish something unpolished, and we hope to ease this inhibition." (Steve Crocker, RFC 3 - 1969)

[23] für eine zuverlässige Internet-Chronologie s. Robert Hobbes Zakon, Hobbes' Internet Timeline, http://www.isoc.org/guest/zakon/Internet/History/HIT.html; eine weitere ausgezeichnete bebilderte Chronologie vom Computer Museum, Moffett Field, California: http://www.computerhistory.org/exhibits/internet_history/

ARPANET und erfand die ‘user@host’-Konvention. Larry Roberts schrieb einen Mail-Client dafür.

Das Netzwerk konnte sich sehen lassen, und so war es Zeit für eine erste öffentliche Demonstration, die 1972 auf der *International Conference on Computer Communications* in Washington stattfand. Im Keller des Konferenzhotels wurde ein Paketvermittlungsrechner und ein *Terminal Interface Processor* (TIP) installiert, der anders als ein IMP den Input von mehreren Hosts oder Terminals verarbeiten konnte. Angeschlossen waren 40 Maschinen in den ganzen USA. Zu den Demonstrationen gehörten interaktive Schachspiele und die Simulation eines Lufverkehrskontrollsystems. Berühmt wurde die Unterhaltung zwischen ELIZA, Joseph Weizenbaum’s künstlich-intelligentem Psychiater am MIT, und PARRY, einem paranoiden Programm von Kenneth Colby an der Stanford Uni. Teilnehmer aus England, Frankreich, Italien und Schweden waren dabei. Vertreter von AT&T besuchten die Konferenz, verließen sie jedoch in tiefer Verwirrung.

Im selben Jahr starteten Projekte für Radio- und Satelliten-gestützte Paketvernetzung, letztere mit Instituten in Norwegen und England. Bob Metcalfe umriß in seiner Doktorarbeit an der Harvard Uni das Konzept für ein *Local Area Network* (LAN) mit multiplen Zugangskanälen, das er Ethernet nannte. Am Xerox PARC entwickelte er das Konzept weiter, bevor er später 3COM gründete.

ARPANET, SATNET und das Radionetz hatten verschiedene Schnittstellen, Paketgrößen, Kennzeichnungen und Übertragungraten, was es schwierig machte, sie untereinander zu verbinden. Bob Kahn, der von BBN an die DARPA ging, und Vint Cerf, der jetzt an der Stanford Uni unterrichtete, begannen, ein Protokoll zu entwickeln, um verschiedene Netze miteinander zu verbinden. Im Herbst 1973 stellten sie auf einem Treffen der *International Network Working Group* in England den ersten Entwurf zum *Transmission Control Protocol* (TCP) vor.

Im Jahr darauf wurde TCP zeitgleich an der Stanford Uni, BBN und dem University College London (Peter Kirstein) implementiert. “So effort at developing the Internet protocols was international from the beginning.” (Cerf[24]) Es folgten vier Iterationen des TCP-Protokollsatzes. Die letzte erschien 1978.

1974 startete BBN Telenet, den ersten öffentlichen paketvermittelten Datenkommunikationsdienst, eine kommerzielle Version des ARPANET. Aufgrund der DARPA-Förderung besaß BBN kein exklusives Recht am Quellcode für die IMPs und TIPs. Andere neue Netzwerkunternehmen forderten BBN auf, ihn freizugeben. BBN sträubte sich zunächst, da der Code ständig verändert würde, doch gab ihn 1975 frei.

Wachstum

Mit der Forschungsförderung für die Implementierung von TCP hatte die DARPA ihre initiale Mission erfüllt. 1975 wurde die Verantwortung für das ARPANET an die *Defense Communications Agency* (später umbenannt in *Defense Information Systems Agency*) übertragen. BBN blieb der Auftragnehmer für den Betrieb des Netzes, doch militärische Sicherheitinteressen wurden jetzt wichtiger. Zusätzlich zur DARPA förderte auch die *National Science Foundation* (NSF) die Forschung in Informatik und Netzwerken an rund 120 US-amerikanischen Universitäten. Weitere Einrichtungen, wie das Energieministerium und die NASA starten eigene Netzwerke. Anfang 1975 verfügte das ARPANET über 61

[24] Cerf 1993

Knoten.[25] Die erste Mailinglist wurde eingerichtet. Zusammen mit den RFCs werden Mailinglisten zum wichtigsten Mittel der offenen Kooperation der technischen *Community*. In der beliebtesten Liste dieser Zeit diskutierte man jedoch über Science-Fiction. Der *Jargon File*, ein Wörterbuch der Hacker-Kultur, zusammengestellt von Raphael Finkel, wurde zum ersten Mal publiziert, natürlich im Netz.

UUCP (Unix to Unix Copy) wurde 1976 an den AT&T Bell Labs entwickelt und als Teil der Unix Version 7 verbreitet. Einrichtungen, die sich keine Standleitung leisten konnten, ermöglichte UUCP, über *Dial-up*-Telefonleitungen Daten mit Rechnern am ARPANET auszutauschen.

Das neue netzwerkverbindende TCP wurde im Juli 1977 erstmals in einem aufwendigen Versuchsaufbau demonstriert. Die Übertragungsstrecke begann mit einem mobilen Paketsender in einem fahrenden Auto auf dem San Francisco Bayshore Freeway, lief zu einem Gateway bei BBN, über das ARPANET, über eine Punkt-zu-Punkt-Satellitenverbindung nach Norwegen, von dort via Landleitung nach London, zurück über das Atlantic Packet Satellite Network (SATNET) ins ARPANET und schließlich zum Informatikinstitut der University of Southern California. “So what we were simulating was someone in a mobile battlefield environment going across a continental network, then across an intercontinental satellite network, and then back into a wireline network to a major computing resource in national headquarters. Since the Defense Department was paying for this, we were looking for demonstrations that would translate to militarily interesting scenarios.”[26]

Seit Mitte der Siebziger wurden Experimente zur paketvermittelten Sprachübertragung durchgeführt. TCP ist auf zuverlässige Übertragung ausgelegt. Pakete, die verloren gehen, werden erneut geschickt. Im Falle von Sprachübertragung ist jedoch der Verlust einiger Pakete weniger nachteilig als eine Verzögerung. Aus diesen Überlegungen heraus wurde 1978 TCP und IP getrennt. IP spezifiziert das *User Datagram Protocol* (UDP), das noch heute zur Sprachübertragung verwendet wird.[27]

Damit wird 1978 das ARPANET-Experiment offiziell beendet. Im Abschlußbericht heißt es “This ARPA program has created no less than a revolution in computer technology and has been one of the most successful projects ever undertaken by ARPA. The full impact of the technical changes set in motion by this project may not be understood for many years.”[28] Einer der Pioniere erinnert sich an die entscheidenden Faktoren:

> “For me, participation in the development of the ARPANET and the Internet protocols has been very exciting. One important reason it worked, I believe, is that there were a lot of very bright people all working more or less in the same direction, led by some very wise people in the funding agency. The result was to create a community of network researchers who believed strongly that collaboration is more

---

[25] Karte s. http://www.computerhistory.org/exhibits/internet_history/full_size_images/1975_net_map.gif

[26] Cerf 1993

[27] Cerf 1993

[28] ARPANET Completion Report, January 3, 1978, zit. nach http://www.computerhistory.org/exhibits/internet_history/internet_history_70s.page

powerful than competition among researchers. I don't think any other model would have gotten us where we are today." (Robert Braden in RFC 1336[29])

Institutionalisierung

Um die Vision eines freien und offenen Netzes fortzuführen, richtete Vint Cerf 1978 noch vom DARPA aus das *Internet Configuration Control Board* (ICCB) unter Vorsitz von Dave Clark am MIT ein. 1983 trat das *Internet Activities Board* (IAB) (nach der Gründung der *Internet Society* umbenannt in *Internet Architecture Board*) an die Stelle des ICCB.

Für die eigentliche Entwicklungsarbeit bildeten sich 1986 unter dem IAB die *Internet Engineering Task Force* (IETF)[30] und die *Internet Research Task Force* (IRTF). Anders als staatliche Standardisierungsgremien oder Industriekonsortien ist die IETF -- "by law and strong custom" -- ein offenes Forum. Mitglied kann jeder werden, indem er eine der etwa hundert aufgabenorientierten Mailinglisten subskribiert und sich an den Diskussionen beteiligt. "In theory, a student voicing a technically valid concern about a protocol will deserve the same careful consideration, or more, as a man from a multibillion-dollar company worrying about the impact on his installed base."[31] Alle Arbeit (mit Ausnahme des Sekretariats) ist unbezahlt und freiwillig.

Die Entwicklungsarbeit innerhalb der IETF gehorcht einem begrenzten Anspruch. Die Ergebnisse müssen ein anstehendes Problem möglichst direkt und, gemäß einer Hacker-Ästhetik von Eleganz, möglichst einfach und kompakt lösen. Sie müssen mit den bestehenden Strukturen zusammenarbeiten und Anschlüsse für mögliche Erweiterungen vorsehen. Da es keine scharf umrissene Mitgliedschaft gibt, werden Entscheidungen nicht durch Abstimmungen getroffen. Das Credo der IETF lautet: "We reject: kings, presidents and voting. We believe in: rough concensus and running code."[32] Wenn sich ein interessantes Problem und genügend Freiwillige finden, wird diskutiert, ablauffähiger Code auch für alternative Lösungsansätze geschrieben und solange getestet, bis sich ein Konsens herausbildet. Wenn dies nicht geschieht, das Verfahren auf unlösbare Probleme stößt oder die Beteiligten das Interesse verlieren, kann ein Standard auch vor seiner Verabschiedung stecken bleiben. Standards und ggf. Code werden in jeder Phase der Entwicklung im bewährten RFC-Format für jeden Interessierten zugänglich veröffentlicht. Das führt dazu, daß sie frühzeitig von einer Vielzahl von Anwendern unter den unterschiedlichsten Bedingungen getestet werden und diese breiten Erfahrungen in den Entwicklungsprozeß eingehen, bevor ein Standard offiziell freigegeben wird. Die Standards sind offen und frei verfügbar. Anders als im ISO-Prozeß können von den an der Standardisierung Beteiligten keine Patente erworben werden, und anders als die ISO finanziert sich die IETF nicht aus dem Verkauf der Dokumentation von Standards. Der kontinuierlichen Weiterentwicklung dieses Wissens steht somit nichts im Wege.

1988 legte der außer Kontrolle geratene Morris-Wurm 6.000 der inzwischen 60.000 Hosts am Internet lahm.[33] Daraufhin bildet die DARPA das *Computer Emergency Response Team* (CERT), um auf zukünftige Zwischenfällen dieser Art reagieren zu können.

---

[29] Malkin 1992

[30] http://www.ietf.org/

[31] Alvestrand 1996: 61

[32] Dave Clark, IETF Credo (1992), http://info.isoc.org:80/standards/index.html

[33] http://www.mmt.bme.hu/~kiss/docs/opsys/worm.html

Die 1990 von Mitch Kapor gegründete *Electronic Frontier Foundation* (EFF) ist keine Internet-Institution im engeren Sinne, doch als Öffentlichkeits- und Lobbyingvereinigung zur Wahrung der Bürgerrechte im Netz hat sie in den USA eine bedeutende Rolle gespielt.

Als Dachorganisation für alle Internet-Interessierten und für die bestehenden Gremien wie IAB und IETF gründeten u.a. Vint Cerf und Bob Kahn 1992 die *Internet Society* (ISOC).[34]

Im Jahr darauf etablierte die NSF das InterNIC (Network Information Center), das bestimmte Dienste in seinem Aufgabenbereich an Dritte ausschrieb, nämlich Directory- und Datenbankdienste an AT&T, Registrierungsdienste an Network Solutions Inc. und Informationsdienste an General Atomics/CERFnet.

Netzwerkforschung

Auf Initiative von Larry Landweber erarbeiteten Vertreter verschiedener Universitäten (darunter Peter Denning und Dave Farber) die Idee eines Informatik-Forschungsnetzes (CSNET). Ein Förderungsantrag an die NSF wurde zunächst als zu kostspielig abgelehnt. Auf einen überarbeiteten Antrag hin bewilligte die NSF 1980 dann fünf Millionen Dollar über einen Zeitraum von fünf Jahren. Das Protokoll, das die verschiedenen Subnetze des CSNET verbindet, ist TCP/IP. 1982 wurde beschlossen, daß alle Systeme auf dem ARPANET von NCP auf TCP/IP übergehen sollen -- obgleich davon nur einige hundert Computer und ein Dutzend Netze betroffen waren, keine einfache Operation (RFC 801).

CSNET und ARPANET wurden 1983 verbunden, doch US-amerikanische Wissenschaftler klagten, daß die Supercomputern des Landes nicht zugänglich seien. Astrophysiker mußten nach Deutschland reisen, um einen in den USA hergestellten Supercomputer verwenden zu können. Im Juli 1983 gab daraufhin eine NSF-Arbeitsgruppe einen Plan für ein *National Computing Environment for Academic Research* heraus, der drei Jahre später in das NSFnet münden sollte.

Die Supercomputer-Krise führte dazu, daß die NSF eine neue Abteilung für *Advanced Scientific Computing* mit einem Etat von 200 Millionen Dollar über fünf Jahre etablierte. Kontrakte für die Einrichtung von Supercomputer-Zentren gingen an verschiedene Universitäten. Diese Zentren wurden durch das NSFNet verbunden, das 1986 mit einem landesweiten 56 Kbps-Backbone startete, der auf Grund des großen Bedarfs bald auf T1 (1,544 Mbps) erweitert wurde. Die Aufträge zur Verwaltung des Backbones gingen an Merit, MCI und IBM, das die Router-Software entwickelte. Um den NSFnet-Backbone herum entwickelten sich eine ganze Reihe NSF-geförderter regionaler Netzwerke. Von Anfang 1986 bis Ende 1987 stieg die Gesamtzahl der Netzwerke am Internet von 2.000 auf beinah 30.000. 1989 nimmt das NSFNet dann den Betrieb von T3-Leitungen (44,736 Mbps) auf. Bob Kahn und Vint Cerf planen bereits ein Versuchsnetz mit 6 Gigabit.

Neue Architekturen, Protokolle und Dienste

Neben TCP/IP wurden weiterhin proprietäre Protokolle eingesetzt (wie DECNet oder NetBEUI von Microsoft und IBM, die es nur Rechnern desselben Herstellers erlauben, miteinander zu sprechen), aber es enstanden auch neue offene Protokolle. Das wichtigste

[34] http://www.isoc.org

darunter ist das BITNET (das *Because It's Time NETwork*), das 1981 als ein kooperatives Netzwerk an der City University of New York startete und die erste Verbindung an die Yale University legte. Zu den Eigentümlichkeiten von BITNET gehört z.B., daß es die Dateiübertragung per eMail realisiert. 1987 überschritt die weltweite Zahl der BITNET-Hosts 1.000.

TCP/IP wurde zum de facto Standard, doch die Anerkennung als offizieller Standard blieb ihm verwehrt. Ein Irrweg in der Netzwerkentwicklung begann, als die *International Standards Organization* (ISO) ab 1982 ein Referenzmodell für einen eigenen verbindungsorientierten Internetzwerk-Standard namens Open Systems Interconnection (OSI) entwickelte. Im Gegensatz zum *bottom-up*-Prozeß der Internet-Community beruht das Standardisierungsverfahren der ISO auf einem vertikalen, mehrschichtigen Prozeß aus Vorschlägen, Ausarbeitungen und Abstimmungen, der zwischen den nationalen Standardisierungsorganisationen, den Arbeitsgruppen und schließlich dem Plenum der ISO hin- und hergeht. Dabei sollen alle Interessen berücksichtigt werden. Der Standard soll in einem theoretischen Sinne vollständig sein. Er soll zugleich rückwärtskompatibel und abstrakt genug sein, um zukünftige Entwicklungen nicht zu verbauen. Durch die begrenzte Zirkulation in den am Verfahren beteiligten Institutionen werden Standards auch nur begrenzt getestet, bevor sie verabschiedet werden. Ist ein Standard endlich verabschiedet, ist er von der Technologie oft genug schon überholt. OSI hat sich nie sehr weit von den Papierkonzepten in den praktischen Computereinsatz hinein entwickelt und gilt heute als gescheitert, doch bis in die Neunziger dekretierten die Forschungs- und Technologiebehörden vieler Ländern, darunter Deutschland und Japan, daß OSI das offizielle und damit das einzige Netzwerkprotokoll sei, in das Forschungsmittel fließen. Selbst die US-Regierung schrieb noch 1988 vor, daß alle Rechner, die für den Einsatz in staatlichen Stellen angekauft werden, OSI unterstützen müssen und erklärte TCP/IP zu einer ‘Übergangslösung’.

1983 beschloß das US-Verteidigungsministerium, das Netz in ein öffentliches ARPANET und das vertrauliche MILNET aufzuteilen. Nur 45 der 113 Host-Rechner blieben im ARPANET übrig. Die Zahl der an diese Hosts angeschlossenen Rechner war natürlich viel größer, vor allem durch den Übergang von *Time-Sharing*-Großrechnern hin zu Workstations in einem Ethernet-LAN. Jon Postel wies den einzelnen miteinander verbundenen Netzen erst Nummern zu, dann entwickelte er zusammen mit Paul Mockapetris und Craig Partridge das *Domain Name System* (DNS),[35] mit einem ersten Name-Server an der University of Wisconsin, der Namen in Nummern übersetzt. Gleichzeitig empfahl er das heute übliche user@host.domain-Adressierungsschema. Das neue Adressensystem institutionalisierte sich 1988 mit der *Internet Assigned Numbers Authority* (IANA), deren Direktor Postel wurde.

1981 begann Bill Joy an der Berkeley University mit einem Forschungsauftrag der DARPA, die TCP/IP-Protokolle in die dort gepflegte freie Version des Betriebssystems Unix zu integrieren. Sie wurden im August 1983 in der BSD-Unix-Version 4.2 veröffentlicht. Die Betriebssysteme von Computer und Netz waren verschmolzen. Nicht zuletzt deshalb begannen viele Computer-Unternehmen, wie z.B. das von Joy mitgegründete Sun Microsystems, BSD zur Basis ihrer Workstations zu machen. Die freie Software 4.2BSD

[35] die generischen *Top-Level Domains* sind .gov, .mil, .edu, .org, .net, .com. und das wenig gebrauchte .int für internationale Organisationen. Domains außerhalb der USA erhalten einen Ländercode aus zwei Buchstaben nach ISO-Norm, z.B. .de, .nl, .jp.

verbreiteten sich rasch. Tausende von Entwicklern in der ganzen Welt übernahmen es und legten so die Grundlage für das heutige globale Internet.

1977 waren mit dem Tandy TRS-80 und dem Commodore Pet die ersten Computer für den Privatgebrauch auf den Markt gekommen und Steve Wozniak und Steve Jobs kündigten den Apple II an. Der IBM-PC folgt 1981 und kurz darauf die ersten IBM PC-Clones. Durch die billigen Kleinstrechner und ihre Fähigkeit, per Modem zu kommunizieren, betritt eine neue Generation von Nutzerkulturen die Informatik- und Netzwelt.

Die Integration von TCP/IP und lokalen Ethernets trieb die Ausbreitung des Internet voran.[36] Ethernet-Karten wurden auch für PCs verfügbar. Anfang der Achtziger entwickelten Studenten von Professor David Clark am MIT, den ersten TCP/IP-Stack für MS-DOS. Der Quellcode für PC/IP und einige einfache Netzapplikationen verbreiteten sich rasch und inspirierte viele andere, den PC für das Internet zu erschließen. Da DOS nicht multitasking-fähig ist, konnte PC/IP nur eine einzige Verbindung (ein *Socket*) unterstützen. Für einige Anwendungen (wie Telnet) stellt die Beschränkung kein Problem dar, FTP dagegen benötigt zwei Verbidungen gleichzeitig, einen Kontroll- und einen Datenkanal. Phil Karn, damals bei den Bell Labs beschäftigt, begann 1985 einen neuen TCP/IP-Stack zu schreiben, bei dem er Multitasking innerhalb der Applikation realisierte, ein waghalsiger Trick, aber er funktionierte. Für CP/M entwickelt, portierte Karn den Code bald auf DOS und, da er ein leidenschaftlicher Amateurfunker ist, überarbeitete er ihn außerdem für die Verwendung über Packet-Radio. Unter dem Namen seines Rufzeichens KA9Q[37] gab er den Code für nichtkommerzielle Verwendung frei.[38]

1979 entstand das USENET, das zu einem Internet-weiten schwarzen Brett werden sollte. Steve Bellovin schrieb dazu einige Shell-Skripte, die es einem Rechner erlauben, über UUCP Nachrichten auf einem anderen Rechner abzurufen. Technisch ist das USENET ein frühes Beispiel für Client-Server-Architekturen. Sozial bildet es einen öffentlichen Raum, in dem jeder lesen und schreiben kann, zu Themen, die so ziemlich alles unter der Sonne umfassen.[39]

Eine andere Form von kooperativem sozialem Raum, der zusätzlich synchrone Kommunikation ermöglicht, sind Multi-User Dungeons (MUD). Als Spielumgebungen entstanden, werden sie später auch für Bildungs- und Diskussionszwecke Verwendung finden. Das erste von ihnen, das MUD1, schrieben ebenfalls 1979 Richard Bartle und Roy Trubshaw an der University of Essex. 1988 kommt mit dem *Internet Relay Chat* (IRC) von Jarkko Oikarinen ein weiteres synchrones Kommunikationsformat hinzu.

Parallel zum Inernet kamen lokale Diskussionsforen, *Bulletin Board Systems* (BBS) auf, zunächst als alleinstehende PCs mit einer oder mehreren Einwahlverbindungen. Mit Hilfe von Telefonleitungen und X.25 vernetzen sich auch diese Kleinrechner, z.B. zum FidoNet, 1983 von Tom Jennings entwickelt. 1985 gründet Stewart Brand das legendäre BBS *Whole Earth 'Lectronic Link* (WELL) in San Francisco. Kommerzielle BBSs wie CompuServe und AOL folgten. Auch diese separaten Netze richten Ende der Achtziger Gateways zum Internet ein, über die sie eMail und News austauschen können (Fidonet z.B. 1988, MCIMail und Compuserve 1989).

---

[36] 1981 brachte Metcalfes 3COM UNET auf den Markt, ein Unix TCP/IP-Produkt, das auf Ethernet läuft.

[37] der Quellcode ist abrufbar unter http://people.qualcomm.com/karn/code/ka9qos/

[38] vgl. Baker 1998

[39] Zur Kultur des USENET siehe ausführlich Hauben/Hauben 1996

Um auch Nicht-Universitätsangehörigen Zugang zum Internet zu ermöglichen, entstanden eine Reihe von Freenets. Das erste, das Cleveland Freenet, wurde 1986 von der Society for Public Access Computing (SoPAC) in Betrieb genommen.

Die Masse der im Internet verfügbaren Informationen wird immer unüberschaubarer. Der Bedarf nach Navigations- und Suchwerkzeugen führte zu neuen Entwicklungen an verschiedenen Forschungseinrichtungen. Am CERN stellte Tim Berners-Lee 1989 Überlegungen zu einem verteilten Hypertext-Netz an, aus dem das *World-Wide Web* (WWW) werden wird. Ähnliche Verknüpfungen bieten die im folgenden Jahr gestarteten Dienste Archie (von Peter Deutsch, Alan Emtage und Bill Heelan, McGill University) und Hytelnet (von Peter Scott, University of Saskatchewan). 1991 kamen *Wide Area Information Servers* (WAIS, von Brewster Kahle, Thinking Machines Corporation) und Gopher (von Paul Lindner und Mark P. McCahill, University of Minnesota) dazu, und die erste Version von Berners-Lees WWW wird freigegeben. Im Jahr darauf entsteht am *National Center for Supercomputing Applications* (NCSA) der erste Web-Browser Mosaic. Ebenfalls 1992 veröffentlich die University of Nevada Veronica, ein Suchwerkzeug für den Gopher-Raum. Im selben Jahr startet der Bibliothekar Rick Gates die *Internet Hunt*, ein Suchspiel nach Informationen, bei dem auch diejenigen aus den veröffentlichten Lösungsstrategien lernen konnten, die sich nicht selber beteiligten.

1990 wurde die erste fernbedienbare Maschine (über SNMP) ans Netz gehängt, der Internet Toaster von John Romkey. Bald folgten Getränkeautomaten, Kaffeemaschinen und eine Fülle von Web-Kameras.

Die offene Architektur des Internet macht es möglich, jede Kommunikation an allen Knoten zwischen Sender und Empfänger abzuhören. Die Antwort darauf lautet Kryptographie, doch die galt als militärisch-staatliches Geheimwissen. Das erste für Normalsterbliche zugängliche Kryptographie-Werkzeug war PGP (Pretty Good Privacy), 1991 von Philip Zimmerman freigegeben.

Neben Texten fanden sich auch schon in den Achtzigern Bilder und Audiodateien im Netz, doch ihre Integration hatte mit dem WWW gerade erst begonnen. Die ersten regelmäßigen ‘Radiosendung’ im Netz waren die Audiodateien des 1993 von Carl Malamud gestarteten *Internet Talk Radio*, Interviews mit Netzpionieren. Weiter ging der Multimedia-Backbone (MBONE), über den 1992 die ersten Audio- und Video-Multicasts ausgestrahlt wurden. Anfangs konnten sich daran nur wenige Labors mit einer sehr hohen Bandbreite beteiligen, doch bald wurden die hier entwickelten Werkzeuge auch für den Hausgebrauch weiterentwickelt. CUSeeMe bot Video-Conferencing für den PC. Das Streaming-Format RealAudio (1995) machte es möglich, Klanginformationen *on demand* und in Echtzeit im Netz abzurufen. Multimediale Inhalte können mit MIME (Multimedia Internet Mail Extensions -- RFC 1437) seit 1993 auch in eMails verschickt werden.

Internationalisierung

In Europa gab es Anfang der Achtziger bereits erste auf Wählverbindungen und UUCP basierende Netze, wie z.B. das 1982 etablierte EUnet (European Unix Network) mit Knoten in Holland, Dänemark, Schweden, und England. In Deutschland kannte man das Internet höchstens aus dem Kino (“War Games”), wie sich einer der deutschen Internet-Pioniere, Claus Kalle vom Rechenzentrum der Universität Köln, erinnert.[40] Großrechner

[40] C. Kalle, Wizards 7/1999

kommunizierten über das teure Datex-P. Das erste Rechnernetz, das über einen eMail-Link in die USA und dort über ein Gateway ins Internet verfügte, war das 1984 gestartete EARN (European Academic Research Network). Natürlich wurde auch bald mit TCP/IP experimentiert -- die RFCs, die man sich per eMail über EARN beschaffen konnte, machten neugierig -- doch das Klima war für TCP/IP nicht günstig. Als 1985 der Verein zur Förderung eines Deutschen Forschungsnetzes e.V. (DFN-Verein) gegründet wurde, vertrat er ausschließlich die OSI-Linie. "In Deutschland und Europa war man damals vollkommen davon überzeugt und förderte auch politisch und finanziell, daß die Protokolle der OSI-Welt in Kürze weit verfügbar und stabil implementiert seien, und damit eine Basis für die herstellerunabhängige Vernetzung existieren würde."[41]

Die ersten Verbindungen von Rechnern außerhalb der USA laufen über UUCP. 1984 wurde z.B. das JUNET (Japan Unix Network) etabliert, und eine erste Botschaft von "Kremvax" sorgte für Aufregung, da seither auch die UdSSR an das USENET angeschlossen war.

Die Inititiative für ein IP-Netz in Deutschland ging 1988 von der Universität Dortmund aus. Es hatte im Rahmen des europaweiten InterEUnet-Verbundes eine Anbindung erst über Datex-P, dann über eine Standleitung nach Amsterdam und von dort aus an das US-amerikanische Internet. Die Informatik-Rechnerbetriebsgruppe (IRB) der Universität Dortmund betrieb einen anonymous-ftp-Server. "Besonders förderlich war es, mit den recht frischen Kopien der GNU- und anderer Public-Domain-Pakete (emacs, gcc, ISODE usw.) zu arbeiten. Auch war auf diesem Wege erstmalig Zugang zu Netnews und Internet-Mail möglich, so daß man sich auf dem Laufenden halten konnte."[42] Eine ähnliche Initiative gab es am Informatik-Lehrstuhl von Professor Zorn an der Universität Karlsruhe, die zum Aufbau des XLINK (eXtended Lokales Informatik Netz Karlsruhe) führte, das ebenfalls eine Verbindung in die USA zum NYSERNet anbot.

Das OSI-Regime des DFN lockerte sich nach und nach. Das X.25-basierte Wissenschaftsnetz (WiN) sollte gleich von seinem Start an auch TCP/IP-Hosts unterstützen.[43] Die europäischen Netzanbieter schlossen sich 1989 auf Initiative von Rob Blokzijl am *National Institute for Nuclear Physics and High-Energy Physics* in Amsterdam zum RIPE (Reseaux IP Europeens) zusammen, um die administrative und technische Koordination für ein paneuropäisches IP-Netzwerk zu gewährleisten. Zur Konsolidierung der bereits existierenden europäischen IP-Netze begannen 1991 einige Netzbetreiber, eine europäische IP-Backbone-Struktur namens EBONE zu planen und aufzubauen.

1992 begannen dann auch Initiativen wie der Individual Network e.V. (IN) mit dem Aufbau alternativer Verfahren und Strukturen zur Bereitstellung von IP-Diensten. Auch das IN nahm im Weiteren aktiv an der Gestaltung der deutschen IP-Landschaft teil. Nicht zuletzt die Netnews-Verteilung wäre ohne die IN-Mitarbeit nur schleppend vorangekommen.

Der Zuwachs der internationalen IP-Konnektivität läßt sich an der Anmeldung von Länder-Domains ablesen. 1988 kamen Kanada, Dänemark, Finland, Frankreich, Island, Norwegen und Schweden dazu. Im November 1989 sind insgesamt 160.000 Hosts am

---

[41] C. Kalle, Wizards 7/1999. Da OSI maßgeblich von Europa und Japan vorangetrieben wurde und als 'Gegenentwicklung' zum US-amerikanischen TCP/IP galt, stand es in einem blockpolitisch aufgeladenen Feld.

[42] C. Kalle, Wizards 7/1999

[43] Zu den Auseindandersetzungen zwischen dem offiziellen OSI-Lager und der wachsenden Zahl von TCP/IP-Verfechtern in Deutschland, s. Kalle, Wizards 7/1999

Internet. Australien, Deutschland, Israel, Italien, Japan, Mexico, Holland, Neuseeland und Großbritannien schließen sich an.

1990 kommen Argentinien, Österreich, Belgien, Brazilien, Chile, Griechenland, Indien, Irland, Südkorea, Spanien und die Schweiz dazu. 1991 sind es Kroatien, die Tschechische Republik, Hong Kong, Ungarn, Polen, Portugal, Singapur, Südafrika, Taiwan und Tunesien. 1992 überschreitet die Zahl der Hosts die eine-Million-Marke. Immer kleinere Länder und Territorien wie Zypern, die Antarktis, Kuwait und Luxemburg melden Länder-Domains an. 1997 kommen noch eine Reihe von Inselnationen und Protektorate hinzu, so daß heute die gesamte Weltkarte auf den Adreßraum des Internet abgebildet ist.

Kommerzialisierung

Das Entstehen eines kommerziellen Internet-Anbietermarktes anzuregen und zu fördern, war eines der Ziele der NSFNet-Initiative. Zu den ersten Nutznießern gehörten Performance Systems International (PSI), Advanced Network and Systems (ANS - von IBM, MERIT und MCI gegründet), Sprintlink und CERFNet von General Atomics, das auch das San Diego Supercomputer Center betrieb. Die kommerziellen ISPs sollten Ende der Achtziger den Erhalt und Ausbau des Internet von den Universitäten und Forschungbehörden übernehmen.

Dadurch entstand auch ein bedeutender Markt für Internet-basierte Produkte. Len Bozack, ein Stanford-Student, gründete Cisco Systems. Andere, wie 3COM, Proteon, Banyan, Wellfleet und Bridge gingen ebenfalls in den Router-Markt.

Die erste Internet-Industriemesse, die Interop in San Jose 1988, zog 50 Aussteller und 5.000 Besucher an.

1991 hob die NSF das bis dahin bestehende Werbeverbot (die *acceptable use policy*) in der öffentlichen Netzinfrastruktur auf. Damit war der Weg frei dafür, daß sich sich General Atomics (CERFnet), PSI (PSInet) und UUNET Technologies, Inc. (AlterNet) in Kalifornien zum ersten CIX (Commercial Internet Exchange) zusammenschlossen, um den ungeeingeschränkten Verkehr zwischen den kommerziellen Netzen zu organisieren.

Auch in Deutschland begann Anfang der Neunziger die Privatisierung der universitären Infrastruktur. Das Drittmittelprojekt EUnet der Informatik-Rechnerbetriebsgruppe der Uni Dortmund wurde Ende 1992 zur GmbH. Im Jahr darauf wurde auch das XLINK-Projekt an der Uni Karlsruhe zur Tochter der NTG, ihrerseits Tochter von Bull.[44]

Wende ab 1990

Ein Wendepunkt läßt sich am Übergang von den 80er zu den 90er Jahren ausmachen. Das ARPANet wird 1990 offiziell abgeschaltet. Die NSF verlagert die Netzwerkförderung von einer direkte Finanzierung der akademischen Backbone-Infrastruktur hin zur Bereitsstellung von Etats, mit denen die Universitäten sich Konnektivität von kommerziellen Anbietern einkaufen.

Mit der schwindenden Rolle der NSF im Internet endete auch die Verbindlichkeit der *Acceptable Use Policy*. Zunächst behutsam, dann in einem unhaufhörlichen Strom setzten die Werbebotschften im Netz ein. Die im CIX zusammengeschalteten Netzanbieter vermarkteten

[44] Kalle, Wizards 7/1999

das Internet als Business-Plattform. Über die Gateways der kommerziellen BBSe kamen Nutzerkulturen, die es gewohnt waren, für Informationen zu bezahlen und ihrerseits die Kanäle hemmungslos für gewerbliche Zwecke zu verwenden. Einen berüchtigten Konflikt löste die Anwaltsfirma Canter & Siegel aus Arizona aus, als sie 1994 Massen-eMails (*Spam*) zur Bewerbung ihrer Green-Card-Lotteriedienste ins Internet schickte. Die Netzbewohner reagierten heftig und unterbanden diesen Mißbrauch, indem sie ihrerseits die Firma massenhaft mit eMail eindeckten.

Ab 1990 wurden gezielte Anstrengungen unternommen, kommerzielle und nichtkommerzielle Informationsdiensteanbieter ins Netz zu holen. Unter den ersten befanden sich Dow Jones, Telebase, Dialog, CARL und die *National Library of Medicine*.

1991 trat das WWW seinen Siegeszug an. Mehr als 100 Länder waren an das Internet angeschlossen, mit über 600.000 Hosts und fast 5.000 einzelnen Netzen. Im Januar 1993 waren es über 1,3 Millionen Rechner und über 10.000 Netzwerke.

US-Präsident Clinton und Vize Al Gore gaben im Februar 1993 unmittelbar nach ihrem Amtsantritt auf einem *Town Meeting* im Silicon Valley eine Erklärung über ihre Technologiepolitik ab, in der das Internet bereits eine zentrale Rolle spielte. Damit lösten sie eine Art Vorbeben aus, in einer geopolitischen Situation, in der die USA sich in einer Wirtschaftskrise befanden, Europa im Aufschwung und Japan an der Weltspitze. Die eigentliche Schockwelle ging über die Welt hinweg, als am 15. September des Jahres Al Gore die *National Information Infrastructure Agenda for Action* verkündete, in der er Netzwerke nicht nur selbst zu einer Multi-Milliarden-Dollar-Industrie, sondern zu einer Grundlageninfrastruktur für Wirtschaft, Bildung, Wissenschaft und Kultur erklärte.[45] Das Bewußtsein, in einem Schlüsseltechnologiesektor hinter den USA herzuhinken, löste allerorten hektisches Treiben aus. Spätestens damit beginnt die kommerzielle Erschließung und die Massenbesiedlung des Internet.

Für die neuen Generationen von Nutzern gibt es nur eine Information, die frei und möglichst weit zirkulieren soll, und das ist Werbung. Alle andere Information ist für sie Ware. Um nun in diesem promiskuitiven Milieu eine Information (z.B. Börsendaten, Lehrmaterial, Musikstücke) derjenigen und nur derjenigen zugänglich zu machen, die dafür bezahlt hat, müssen in das Internet zusätzliche, aufwendige Schutzmechanismen, Zonen mit Zugangskontrollen und kryptographisch abgesicherte *Copyrights Control Systems* eingezogen werden. Die sog. Rechteindustrie (Bertelsmann, Sony, Time-Warner usw.) arbeitet seit etwa 1994 nach Kräften daran, ihre Waren über das Netz verkaufbar zu machen und technisch abzusichern.[46] Nichts demonstriert die neue Qualität des Internet besser, als die erste Cyber-Bank, First Virtual, die 1994 ihren Betrieb aufnimmt.

Microsoft hatte das Internet zunächst verschlafen. Bill Gates erwähnte in der Erstausgabe seines 1995 erschienen Buches “The Road Ahead” das Internet mit keinem Wort. Kurz darauf schwenkte er den Ozeanriesen Microsoft auf Internet-Kurs. Noch im selben Jahr erschien die erste Version des MS Internet Explorers. Nachdem die Kopplung von Hard- und Software gebrochen war, löste das Web die Verbindung von jeweils spezifischer Software und Information auf. Microsoft Network (MSN) war dagegen ein Versuch, erneut eine solche Kopplung zu legen: ein geschlossenes Format, in dem Firmen kostenpflichtige Informationen und Dienstleistungen anbieten konnten -- sofern sie eine

[45] Im selben Jahr richtete die US-Regierung als erste einen Web-Server ein (http://www.whitehouse.gov/).

[46] vgl. Stefik 1996

Startgebühr von $ 50.000 und einen Anteil aller Einnahmen an MS zahlte. Es handete sich um eine verspätete Immitation der geschlossenen BBSe wie Compuserve oder AOL, die bereits durch das WWW überholt waren, das es jedem erlaubte, gebührenfrei Informationen anzubieten.

Domain-Namen waren bislang nichts als eine Mnemotechnik gewesen, die die darunterliegenden numerischen IP-Adressen handhabbarer machten. Durch den Einzug großer Unternehmen mit ihren geschützten Warenzeichen werden sie zu einem aggressiv umstrittenen Territorium. Der erste prominente Streit darüber, ob Domain-Namen geistiges Eigentum sind, war MTV Networks gegen Adam Curry. Etwa im Mai 1993 hatte Curry, ein MTV-Video-Jockey, auf eigene Faust und Kosten ein Informationsangebot unter mtv.com gestartet. In Gesprächen mit führenden Angestellten von MTVN und deren Mutterfirma Viacom New Media hieß es, MTV habe kein Interesse am Internet, hindere ihn aber auch nicht an seinen Aktivitäten. Also baute Curry sein Informationsangebot weiter aus, u.a. mit einem Schwarzen Brett, auf dem sich Musiker und Vertreter der Musikindustrie miteinander unterhielten. In den von ihm moderierten Fernsehprogrammen wurden eMail-Adressen wie "popquiz@mtv.com" eingeblendet. Im Januar 1994 forderte MTVN Curry förmlich auf, die Verwendung von mtv.com einzustellen. Dennoch verwiesen MTV-Sendungen weiterhin auf diese Adresse und ein führender Angestellter bat Curry im Februar, bestimmte Informationen in seiner Site aufzunehmen. Inzwischen hatten MTVN und AOL einen Vertrag abgeschlossen, um einen kostenpflichtigen Dienst anzubieten, der u.a. ein Schwarzes Brett für Musik-Profis beinhalten sollte, das dem von Curry auffällig glich. MTVN verklagte Curry u.a. wegen Verstoßes gegen Trademark-Ansprüche auf Freigabe der Domain mtv.com. Currys Versuche, den Streit gütlich beizulegen, scheiterten und er kündigte. Letztzendlich kam es doch zu einer außergerichtlichen Einigung, bei der Curry mtv.com an MTV aufgab.[47] Die Situation war typisch für die Zeit um 1993-94: große Unternehmen, auch aus der Medienbranche, ignorierten oder unterschätzten die Bedeutung des Internet, während innovative Einzelpersonen durch ihr persönliches Engagement populäre und kostenlose Informationsangebote aufbauten, nur um zusehen zu müssen, wie ihnen ihre Arbeit vom Rechtssystem abgesprochen wird. Nachdem in zahlreichen Urteilen entschieden war, daß Domain-Namen dem Warenzeichenregime unterliegen, setzte ein reger Handel ein. CNET beispielsweise kaufte 1996 die URL 'tv.com' $ 15.000. 'business.com' wurde 1997 für $ 150.000 verkauft und zwei Jahre später für bereits $ 7,5 Millionen weiterverkauft.

Bis 1995 ist die kommerzielle Backbone-Infrastruktur in den USA soweit errichtet und untereinander verschaltet, daß der NSFNET-Backbone-Dienst eingestellt werden kann.[48] Im selben Jahr gehen eine Reihe von Internet-Unternehmen an die Börse, am spektakulärsten das auf der NCSA-Browser-Technologie errichtete Netscape mit dem drittgrößten NASDAQ-IPO-Wert aller Zeiten.

Im Gefolge der Wirtschaft halten auch die Rechtsanwälte Einzug ins Internet. Als Teil der Verrechtlichung unternimmt auch der Gesetzgeber Schritte zu seiner Regulierung. 1996 wird in den USA der umstrittene *Communications Decency Act* (CDA) verabschiedet,

---

[47] Offener Brief von Adam Curry auf der "'Lectric Law Library"-Liste vom 10. Mai 1994, http://www.lectlaw.com/files/inp10.htm; MTV v. Adam Curry case from 867 F.Supp. 202., United States District Court, S.D. New York, Oct. 28, 1994; http://www.loundy.com/CASES/MTV_v_Curry.html

[48] Merit's History. The NSFNET Backbone Project, 1987 - 1995, http://www.merit.edu/merit/archive/nsfnet/transition/

der den Gebrauch von ‘unanständigen’ Wörtern im Internet verbietet. Einige Monate später verhängt ein Gericht eine einstweilige Verfügung gegen die Anwendung dieses Gesetzes. 1997 erklärt das höchste US-Gericht den CDA für verfassungswidrig. Dennoch wird in dieser Zeit der vermeintlich rechtsfreie Raum des Internet in die gesetzlichen Regularien von Kryptografie über Urheberrecht bis zu den Allgemeinen Geschäftsbedingungen einbezogen.

In vielen Ländern greifen Gerichte und staatliche Behörden in den Cyberspace ein. China verlangt, daß ISPs und Nutzer sich bei der Polizei registrieren. Ein deutsches Gericht entscheidet, daß Compuserve den Zugang zu Newsgroups, die sich im weitesten Sinne mit Sexualität beschäftigen, unterbinden muß. Da Compuserve sein weltweites Informationsangebot in seiner Zentrale in Ohio vorrätig hält und es technisch nicht nach einzelnen Länder differenzieren kann, schaltet es die Newsgroups für alle Nutzer ab, was eine vor allem amerikanische Protest- und Boykottwelle gegen Deutschland auslöst. Saudi Arabien beschränkt den Zugang zum Internet auf Universitäten und Krankenhäuser. Singapur verpflichtet politische und religiöse Inhalteanbieter, sich staatlich registrieren zu lassen. Neuseeland klassifiziert Computer-Disketten als ‘Publikationen’, die zensiert und beschlagnahmt werden können. Amerikanische Telekommunikationsunternehmen nehmen Anstoß an Internet-Telefoniediensten und fordern das Parlament auf, die Technologie zu verbieten.

Auch die Selbstorganisation der technischen Entwicklung der Internet-Grundlagen verändert ihren Charakter. Saßen in den jährlichen Treffen von IETF-Arbeitsgruppen Mitte der Achtziger höchstens 100 Personen, sind es jetzt nicht selten 2-3.000. Entsprechend sind sie kein kollektives Brainstorming mehr, sondern dichtgedrängte Abfolgen von Präsentationen. Die eigentliche Arbeit findet immer häufiger in kleinen geschlossenen Gruppen, den Design-Teams, statt. Während die mehr als zwanzig Jahre alte Technologie des Internet erstaunlich stabil skaliert, stoßen die *Community*-Strukturen an ihre Grenzen. Auch die Zusammensetzung der Arbeitsgruppen verändert sich. “Seit den späten 80er Jahren hat sich der Anteil akademischer Mitglieder in der IETF stetig verringert -- und das nicht nur, weil die Zahl der Unternehmen immer mehr anstieg, sondern auch, weil immer mehr Gründungsmitglieder in die Wirtschaft wechselten.”[49] Das kollektive Streben nach der besten Lösung für das Internet als ganzes, so Jeanette Hofmann, droht, von den Interessen konkurrierender Unternehmen unterlaufen zu werden, die ihre jeweiligen Produkte durchsetzen wollen. Schließlich führen die schiere Größe, die nachrückende Generation von Ingenieuren und das Gewicht der gewachsenen Struktur dazu, daß die Standardentwicklung dazu neigt, konservativer und mittelmäßiger zu werden. Hofmanns Fazit: Die IETF sei auf dem besten Weg, eine Standardisierungsorganisation wie jede andere zu werden. “Das Internet und seine Gemeinde sind in der Normalität angekommen. Irgendwann werden sich die Väter unter der wachsenden Zahl gleichberechtigter Mitglieder verloren haben -- und mit ihnen ein Teil der Ideen und Prinzipien, die die Entstehung des Internet umgaben.”[50]

The Beginning of the Great Conversation

Welche Bedeutung hat nun die hier geschilderte Entwicklung des Internet für die Wissensordnung digitaler Medien allgemein und für die freie Software im besonderen? Das

[49] Jeanette Hofmann, Wizards 7/1999

[50] Jeanette Hofmann, Wizards 7/1999

Netz der Netze ist offen, verteilt, dezentral und heterogen. Im Gegensatz zum zentralistischen Telefonsystem beruht es auf lokalen *Peering*-Abkommen zwischen geographisch benachbarten Netzen. Das Internet ist offen, weil es von Beginn an auf den verschiedensten Rechnerarchitekturen implementiert wurde, weil es auf jede Netzwerk-Technologie (Radio, Satelliten, ISDN, Frame Relay, ATM) aufsetzen kann,[51] und weil es seinerseits andere Protokolle (AppleTalk, Novell IPX, DECNet) gekapselt transportieren kann. Offen ist es auch von Beginn an für internationale Zusammenarbeit. Offen ist es schließlich, weil die öffentliche Förderung (durch die US-Wissenschaftsbehörden ARPA und NSF und mit Verzögerung auch durch die entsprechenden Behörden anderer Länder) eine proprietäre Schließung der Forschungsergebnisse verhinderte. Der antikommerzielle Geist in dieser öffentlichen Infrastruktur der Wissenschaftsgemeinde wurde in der *Acceptable Use Policy* der NSF kodifiziert, die bis Ende der achtziger Jahre jegliche kommerzielle Nutzung, wie Werbung oder Verkauf von Waren und Dienstleistungen im Internet, untersagte. Es ging um Grundlagenforschung, in einem Gebiet, in dem viele Leute kooperieren mußten, durchgeführt an Universitäten und in seinem ersten Jahrzehnt noch ohne militärische[52] und wirtschaftliche Anwendung. Aus all diesen Faktoren erwuchs eine kooperative *Community*, die das Netz trägt und von ihm getragen wird.

Aus dieser Zeit und aus diesem Wissensmilieu stammt der Grundsatz der Wissensordnung digitaler Medien, den einige Unverdrossene auch heute im Zeitalter der *dot-com-economy* aufrecht erhalten: *Information wants to be free*. Die besondere Qualität dieser Umgebung ergibt sich aus der Zweiwegkommunikationsstruktur und der Archivfunktion des Netzes. Die schlichte Tatsache, daß jeder Teilnehmer einzelne oder Gruppen beliebiger Größe von anderen Teilnehmern ansprechen kann und daß viele der öffentlichen Äußerungen nachlesbar und referenzierbar bleiben, führt zu dem was John Perry Barlow als “The End of Broadcast Media and the Beginning of the Great Conversation” bezeichnet hat. “If it is suddenly possible to spread ideas widely without first shoving them through some centrally operated and intensely capitalized industrial engine -- whether a complex of book binderies or 50,000 watt transmitters -- ... freedom of expression will belong not only to those who buy ink by the barrel or transmitter power by the kilowatt. No longer will anyone have to confine their expressions to ideas congenial to the media lords and their advertisers.”[53]

Kommunikationsmittel, die in vergleichbarer Form nur großen Unternehmen und gutbesoldeten staatlichen Stellen verfügbar waren (z.B. Video-Conferencing), erlauben es jetzt Individuen, ad hoc oder kontinuierlich zusammenzuarbeiten. Daraus ergeben sich für einzelne Gruppen wie für das Internet insgesamt Strukturcharakteristika, die Helmut Spinner für die vielversprechenste nichttechnische Neuerung der aktuellen Wissensordnung hält. “Es sind die weiten aber trotzdem flachen Informationsnetze, die erstmals in der Geschichte das traditionelle ‘Organisationsgesetz’ durchbrechen, demzufolge Größenwachstum unvermeidlich mit Abschließung nach außen und Hierarchiebildung im Innern verbunden ist, vom Sportverein über den Wirtschaftsbetrieb bis zum Großreich.”[54]

---

[51] Eines der Hauptziele des Projekts war, wie Cerf es beschreibt, “IP on everything.” (Cerf 1993)

[52] Erst Ende 1978 begannen sich die operativen Streikräfte dafür zu interessieren. “In 1979 we deployed packet radio systems at Fort Bragg, and they were used in field exercises... In 1980, it was decided that TCP/IP would be the preferred military protocols.” (Cerf 1993)

[53] Barlow 1996

[54] Spinner 1998: 9

Das Internet entwickelt sich selbst im Modus der offenen Kooperation und wird zur Möglichkeitsbedingung für eine inkrementelle Wissensentwicklung durch Tausende auf der ganzen Welt verteilter Individuen, ohne Management- und andere Overhead-Kosten, in einer direkten Rückopplungsschleife mit den Anwendern. Damit ist es auch die essentielle Möglichkeitsbedingung für das Phänomen der freien Software.

Geschichte der Software-Entwicklung

Der Wandel der Wissensordnung im 20. Jahrhundert wird durch drei Dynamiken charakterisiert: "Technisierung des Wissens, Kommerzialisierung der Wissensdienste, Globalisierung der Rahmenbedingungen unter dem Ordnungsregime der Wirtschaftsordnung".[55] Eine zentrale Rolle darin spielt der Computer. "Als Buchdruck und Nationalstaat die Medientechniken der mittelalterlichen Universität schluckten, blieben die Inhalte des Wissens ziemlich unberührt. Die Speicherung und Übertragung wurden zwar privatisiert oder verstaatlicht, aber die eigentliche Verarbeitung des Wissens lief weiter in jenem schönen alten Rückkopplungskreis aus Auge, Ohr und Schreibhand. Genau das hat die Computerrevolution verändert. Universale Turing-Maschinen machen auch und gerade die Verarbeitung des Wissens technisch reproduzierbar."[56]

In den frühen Tagen des Computers war alle Software quelloffen. Auch in der kommerziellen Computer-Welt waren damals die Quellen verfügbar, einfach weil Software noch keinen eigenständigen Markt darstellte. Die Hardware-Hersteller lieferten sie gleichsam als Gebrauchsanweisung dazu, alles weitere schrieben die Anwender selbst. Die Computer-Unternehmen hatten es also mit programmierkompetenten Nutzern zu tun, und förderten deren Selbstorganisation und gegenseitige Unterstützung in *User-Groups* wie IBMs SHARE[57] und DECs DECUS. Auch in Zeitschriften wie in der Algorithmen-Rubrik der *Communications of the ACM* oder im Hobbybereich in Amateurfunkzeitschriften zirkulierte uneingeschränkter Quellcode. Entwickler bauten auf den wachsenen Bestand der vorhandenen Software, verbesserten sie, entwickelten sie weiter und gaben ihre Ergebnisse wieder zurück an die Nutzergemeinschaft. Bezahlt wurden sie, ob in Universität oder Firma, für das Programmieren, nicht für die Programme.

Auch an den Universitäten war es bis in die siebziger Jahre üblich, die dort entwickelte Software frei zu verbreiten, so hielt es z.B. das MIT mit dem PDP-6/10 Assembler HACKMEM oder dem ursprünglichen Window-System und Donald E. Knuth von der Stanford Universität mit seinem Satzsystem TeX (1978).

In den sechziger Jahren dominierten Hardware-Firmen, wie IBM, DEC, Hewlett-Packard und Data General, die Computer-Industrie. Eine Unterscheidung in Hardware- und Software-Unternehmen existierte noch nicht. Beim Marktführer IBM konnten Kunden Hardware nur in einem Paket kaufen, das Software, Peripheriegeräten, Wartung und Schulung einschloß. Dadurch hatten kleinere Wettbewerber kaum eine Chance, diese Leistungen anzubieten. 1969 begann IBM dieses *Bundling* aufzugeben. Die offizielle Firmengeschichte von IBM bezeichnet diesen Schritt euphemistisch als *"innovations in*

---

[55] Spinner 1998: 34

[56] Kittler, Wizards 7/1999

[57] 1955 gegründet, ist SHARE auch heute noch aktiv, s. http://www.share.org

marketing"[58] Tatsächlich erfolgte er unter dem Druck des gerade vom US-Justizministerium eingeleiteten Kartellverfahrens.[59] Das 'freiwillige' *Unbundling* beinhaltete u.a., daß ein Teil der Software als separate Produkte angeboten wurde. Eine große Zahl von Programmen stand jedoch frei zur Verfügung, da IBM sie in der *public domain* erachtete. Da sie aus der Nutzer-Community stammten, hätte das Unternehmen auch kaum ein Urheberrecht darauf beanspruchen können. Dieser Schritt zog weitere Kritik auf sich, da natürlich niemand eine Software-Firma dafür bezahlen würde, die gleichen Programme zu schreiben.[60]

Trotz der Konkurrenz durch freie Software, schuf diese Entkopplung die Voraussetzung für eine eigenständige Software-Industrie, auch wenn die in den Siebzigern neu entstandenen Software-Firmen meist noch Satelliten von Hardware-Herstellern waren. Software war erstmals zu einer Ware geworden. Eine berüchtigte Episode ist 1976 der "Open Letter to Fellow Hobbyists".[61] Bill Gates beklagte darin, daß die meisten seiner Fellows ihre Software 'stehlen' würden, was verhindere, daß gute Software geschrieben werden könne. Der neue Zeitgeist ist umso bemerkenswerter, als Gates kurz vorher beinahe aus der Harvard Universität geflogen wäre, weil er die öffentlich finanzierten Ressourcen mißbraucht hatte, um kommerzielle Software zu schreiben. Nachdem er gezwungen worden war, seine Software in die Public Domain zu stellen, verließ er Harvard und gründete Microsoft.

Den überwiegenden Teil stellte jedoch weiter maßgefertigte Software dar, die in den EDV-Abteilungen oder bei externen Dienstleistern geschrieben wurde. Es entstanden Tausende von nahezu identischen Buchhaltungs-, Abrechnungs- und Datenbanksystemen, immer aufs neue. Am Beginn von Software als Massenware aus dem Regal stand eine weitere folgenreiche Entscheidung von IBM.

Anfang der achtziger Jahre brachte das Unternehmen den IBM-PC heraus und nahm ihn gleichzeitig nicht ernst. Erst nach einigen Überzeugungsversuchen und Provokationen (Mitarbeiter warfen IBM vor, nicht in der Lage zu sein, einen so kleinen Computer zu bauen) gab die Firmenleitung ihre Skepsis auf und der Entwicklergruppe um Don Estridge in Boca Raton den Auftrag, einen Personal Computer zu entwickeln. Statt wie bislang alle Komponenten im eigenen Haus zu fertigen, kaufte IBM die Bestandteile wie Prozessor und Betriebssystem von außen zu. Die CPU kam von Intel (8088), das Betriebssystem DOS (Disk Operating System) von einer 32-köpfigen Firma namens Microsoft. Einen tiefgreifenden Wandel der Computer-Landschaft bewirkt ferner IBMs Entscheidung, die Spezifikation der Hardware-Architektur des IBM-PC zu veröffentlichen. Ob es ebenfalls an der Verkennung des PC lag, an einem Zufall, einer subversiven Aktion oder an einer glücklichen Fügung des Schicksals,[62] auf jeden Fall hat IBM mit der Veröffentlichung sich selbst Konkurrenz und den

---

[58] http://www.ibm.com/ibm/history/story/text.html

[59] Das Justizministerium wollte IBM, das damals mit einem 70%-Anteil und $ 7 Mrd. Umsatz den Computermarkt dominierte, in sieben Unternehmen aufteilen. Das Verfahre kam nie zu einem Abschluß. Unter der Reagan-Regierung wurde es 1981 eingestellt.

[60] "Prior to the unbundling, one software company actually tried to obtain an injunction prohibiting IBM from providing for its customers a free sorting program similar to one the plaintiff rented for $200 per month. The judge denied the injunction, but not because IBM had a right to give away its programs, nor because the injunction would force users to pay for something they could have had for free. His reason was that the evidence did not show that IBM provided the program mainly to take business away from the other firm, or that the firm was damaged irreparably." Baase 1974

[61] http://www.eskimo.com/~matth/hobby.html

[62] "And that was a decision by IBM that was part accident, part inspiration by some rebels in the organization who kind of slipped it by their top management, partly perhaps a stroke of genius, partly perhaps

Computer zur Massenware gemacht. Ein Industriestandard, unabhängig von einem einzelnen Unternehmen war gesetzt. Eine Fülle neuer Hardware-Hersteller aus West und Fernost unterboten sich mit ihren Preisen für IBM-PC-*Clones*. Leute wie Michael Dell begriffen, daß es nicht mehr vorrangig um die Erfindung neuer Rechnerarchitekturen ging, sondern um Verkauf, Marketing und Distribution von etwas, das zu einer *Commodity* geworden war.[63] Computer wurden billig genug für den Privatgebrauch. Die Branche zollt ihrem ehemaligen Führer Ehre, indem noch jahrelang Intel-Rechner aller Hersteller dieser Welt als 'IBM-Kompatible' bezeichnet wurden. Der eine große Gewinner an diesem Wendepunkt war Intel, da jeder PC mit der offenen, IBM-kompatiblen Architektur einen seiner Prozessoren enthielt.

Der andere große Gewinner war Microsoft. Wie so oft spielte dabei der Zufall eine Rolle. Das am weitesten verbreitete Betriebssystem für die damalige Generation von 8-Bit-Rechnern war CP/M. 1973 hatte Gary Kildall begonnen, einen PL-1 Compiler auf dem Intel-8080 zu implementieren und einige kleinere Routinen zum 'Control Program for Microcomputers' (CP/M) zusammenzufassen.[64] 1975 wurde CP/M, das jetzt auch auf dem Intel-8085 und Zilog-80 lief, erstmals von Digital Research auf dem Markt angeboten. Es wurde weltweit schätzungsweise 500.000 mal installiert und bot ein umfangreiches Angebot an Anwendungsprogrammen diverser Softwarefirmen, darunter das am weitesten verbreitete Textverarbeitungssystem WordStar.[65] Als sich IBM 1980 nach einem Betriebssystem für den PC umsah, war es naheliegend, daß sie sich an Digital Research wandten. Anekdoten erzählen, daß Kildall an diesem Tag mit seinem Privatflugzeug unterwegs war und dadurch das Millionengeschäft mit IBM verlor.[66] Das ging stattdessen an Microsoft, das damals vor allem für das auf fast jedem Mikrocomputer verfügbare MS-BASIC bekannt war. Wie viele seiner Produkte hatte Microsoft MS-DOS nicht selbst entwickelt, sondern von einer Firma namens Seattle Computer gekauft. Es handelte sich um eine abgespeckte Version von CP/M, von der Digital Research noch dazu bewehrte, daß Seattle Computer ihren Quellcode gestohlen habe. Die Industrie war sich einig, daß CP/M ein weit überlegenes Betriebssystem war, doch mit der Unterstützung durch IBM, das sicherstellte, daß alle Programme, die anfänglich für den IBM-PC ausgeliefert wurden, nur mit MS-DOS, nicht aber mit DRs CP/M kompatibel waren, dominierte MS-DOS in kürzester Zeit den Markt. “Microsoft had already mastered the art of leveraging hardware deals, the bundling of applications software and operating systems in a mutually reinforcing strategy to knock off rivals, even when those rivals had superior products.”[67] Im neuen Zeitalter des Desktop-Computers bildete, wie IBM schmerzvoll feststellen mußte, die Kontrolle über das Betriebssystem den Schlüssel zur Etablierung eines Imperiums. Microsoft erbte gleichsam IBMs Monopol, das in derselben Bewegung zuende ging.

---

an accident.” Tim O’Reilly, Wizards 7/1999

[63] Dell/Fredman 1999

[64] Fritsch 1992

[65] COMPUTERWOCHE Nr. 39 vom 21.09.1984, http://www.computerwoche.de/archiv/1984/39/8439c096.html. CP/M lief auf Rechnern wie denen von Zenith, Kaypro, Osborne, Xerox, Vector Graphics, NorthStar, IMSAI, Commodore, Amstrad, Sharp und Starlet.

[66] Fritsch 1992

[67] Newman 1997

Digital Research versuchte noch gut zehn Jahre, durch Innovationen mit MS zu konkurrieren.[68] Während die Kundschaft MS vorwarf, wenig für die Weiterentwicklung von MS-DOS zu tun, waren spätestens 1984 CP/M-Versionen *Multitasking*- und *Multiuser*-fähig. Als Digital Researchs CP/M-Weiterentwicklung DR-DOS einen deutlichen Marktanteil auf dem Desktop errang, verlautbarte MS 1990, daß das *Release* von MS-DOS 5.0 unmittelbar bevorstehe. Der Nachfolger des 1986 erschienen MS-DOS 3.3 sollte alle Features enthalten, die Anwender an DR-DOS schätzten. Es dauert noch mehr als ein Jahr, bis MS-DOS 5.0 tatsächlich auf den Markt kam, doch bremste allein die Ankündigung den Absatz von DR-DOS. Als es schließlich erschien, hatte sich MS ein neues Marketing-Verfahren einfallen lassen, das DR und alle anderen PC-Betriebssystemhersteller endgültig aus dem Markt drängte.[69] MS verlangte von allen Hardware-Verkäufern, die MS-DOS auf ihren Rechnern installieren wollten, daß sie eine Lizenzegebühr für alle ihre Maschinen an MS bezahlten, auch wenn sie einige davon mit anderen Betriebssystemen auslieferten.[70]

Die Möglichkeiten zur Integration erweiterten sich mit MS-Windows, der graphischen Benutzeroberfläche über DOS. Hauptkonkurrent hier war Apples Mac, der diese neue Art der Computer-Bedienung am Markt eingeführt hatte, aber auch Window-Umgebungen auf DOS, wie Quarterdecks Desqview und Digital Researchs Graphics Environment Manager (GEM). 1983 führte MS den Interface Manager erstmals öffentlich vor, ausgeliefert wurde er als Windows 1.0 erst zwei Jahre später. Im Monat der Auslieferung unterzeichneten Apple und MS eine Vereinbarung über die Verwendung von Apples Copyrights auf das Grafik-Display des Mac durch MS. Knapp drei Jahre später verklagt Apple MS wegen Urheberrechtsverletzung in Bezug auf Windows 2.03. Und noch ein Jahr später wird Apple seinerseits verklagt. Xerox behauptete, die graphische Benutzeroberfläche von Lisa und Mac sei eine Kopie des Interfaces von Xerox' Star-System.

Auf der Intel-Plattform war es gleichsam ein Heimspiel für MS. Es verwendete nichtveröffentliches Wissen über die Arbeitsweise von MS-DOS und MS-Windows, um Wettbewerber auszuschließen. So gaben frühe Versionen von MS-Windows falsche Fehlermeldungen aus, die suggerierten, daß es imkompatibel zu DR-DOS sei. Quarterdeck mußte feststellen, daß MS-Anwenderprogramme Systemaufrufe verwendeten, die Desqview nicht anbieten konnte, da sie nirgends dokumentiert waren.[71]

Selbst mit dem ehemaligen Bündnispartner IBM nahm MS es auf. 1987 brachten MS und IBM die 1.0-Version des gemeinsam entwickelten *Multitasking*-fähigen graphischen Betriebssystems OS/2 auf den Markt, das für höherwertige PCs an die Stelle von DOS/Windows treten sollte. 1990 stellen MS und IBM die gemeinsame Arbeit an Betriebssystemen ein. MS konzentriert sich auf das im selben Jahr vorgelegte Windows 3.0.

---

[68] bis 1984 waren erschienen: CP/M-86 für Intels 16-Bit-Prozessoren, CP/M-68 für Motorolas MC 68000, das Multitasking-fähige Concurrent CP/M-86 und die Mehrplatz-Betriebssysteme MP/M und CP/Net. Hinzu kamen *Clones* von Konkurrenten wie CDOS von Cromenco oder KOS von Kontron. (COMPUTERWOCHE Nr. 39 vom 21.09.1984)

[69] DRI wurde schließlich von Novell aufgekauft und dann von Caldera übernommen, das derzeit die Urheberrecht für alle DRI-Software besitzt. Im September 1996 veröffentlichte Caldera den Quellcode für alle DR-Produkte. Eine Sammlung "aufgegebener" kommerzieller Software ohne Quelle unter http://deltasoft.fife.wa.us/cpm. (CP/M-FAQ maintained von Donald C. Kirkpatrick, http://www.psyber.com/%7Etcj/)

[70] Newman 1997

[71] 1992 veröffentlichte Andrew Schulman das Buch "Undocumented Windows", mit Features, die er entdeckte hatte, indem er den Windows-Code analysierte.

IBM versuchte noch einige Jahre, OS/2 gegenüber MS-Betriebssystemen zu plazieren.[72] MS änderte 1991 den Namen des Projekts OS/2 v3.0 in Windows NT.

Auf der Grundlage seiner Betriebssysteme DOS, Windows[73] und NT schuf MS neue Formen von *Bundling*. Es konnte seine Applikationen wie Word, Excel, SQL, PowerPoint[74] und Quicken[75] als Standards auf Intel-Rechnern durchsetzen. Kombinationen dieser Anwendungen brachte MS in den Software-Suiten MS-Works (für DOS 1987) und MS-Office auf den Markt, die zuverlässig untereinander Daten austauschen konnten. Mit Hilfe von proprietären Dateiformaten verhinderte MS den Datenaustausch mit Programmen anderer Hersteller. Zwar schuf die neue Plattform auch Chancen für andere Software-Anbieter, doch bis WordPerfect, Lotus oder Corel ihre Produkte auf die jeweils neueste Version von Windows angepaßt hatten, hatte MS den Markt bereits in der Tasche. Ganze Kategorien von Software-Werkzeugen, wie Dateimanager und Kompressionspgrogramme, die zuvor separat angeboten worden waren, integrierte MS in Windows und trocknete so effektiv den Markt für alternative Produkte aus. Diese Strategie der Integration in Windows setzte MS mit Software für Sprach- und Handschrifterkennung und dem Web-Browser MS-Internet Explorer fort. Durch ein System von Lehrmaterialien, Schulung, Zertifizierung, Entwicklerwerkzeugen, abgestufter Freigabe von programmierrelevanten Informationen und Sonderkonditionen für Universitäten sicherte sich MS die Kontrolle über die Welt der Entwickler, sowohl für die Privat- wie für den Geschäftsbereich.[76] Ausgehend von Betriebssystemen und Applikationen erweitert MS sein Imperium um *Content* (mit Encarta, der ersten Multimedia-Encyclopädie für den Computer, der Allianz mit NBC und seinen Investitionen in den Hollywood Film- und Musikproduzenten Dreamworks), Finanz- und Handelsdienstleistungen (durch Allianzen mit Banken, den online-Verkauf von Flugtickets, Autos, Nachrichten und Unterhaltung) und Netzkonnektivität (mit Microsoft Network, dem Aufkauf von WebTV, den Investitionen in die Kabelunternehmen Comcast und US West und Bill Gates' Milliardenprojekt eines Satellitennetzes namens Teledesic). Die monopolistischen Praktiken von MS waren immer wieder Gegenstand von Kartellverfahren u.a. des US-Justizministeriums und der Europäischen Kommission.

Der Machtwechsel von IBM zu Microsoft stellte eine Verschiebung von Hardware zu Software dar. Hardware wurde zur Massenware, Software wurde zum ersten Mal überhaupt

---

[72] 1993 lag laut MS die Zahl der lizenzierten Windows-Nutzer bei 25 Millionen. Im selben Jahr meldete IBM einen Jahresverlust von 8.1 Milliarden US$ -- etwa so hoch waren MSs Gewinne. (Rice/Pecot o.J.)

[73] Die wachsende Marktmacht läßt sich an den Absatzzahlen ablesen. 1987 wurde die Millionste Kopie des zwei Jahre zuvor erschienen Windows verkauft. Als 1992 Windows 3.1. erschien, wurde die Millionengrenze schon nach den ersten 50 Tage überschritten, beim MS-DOS 6.0 Upgrade von 1993 nach nur 40 Tagen, und als kurz vor Ende des Jahres 1995 Windows 95 erschien, wurden eine Million Kopien innerhalb der ersten vier Tage verkauft. (Rice/Pecot o.J.)

[74] ein Präsentationsgrafikprogramm, das MS 1987 zusammen mit seinem Hersteller Forethought einkaufte.

[75] eine Finanz-Software, die MS 1994 zusammen mit ihrem Hersteller Intuit aufkaufte.

[76] 1997 verwendeten 65% aller Software-Entwickler weltweit MS-Produkte. 2,4 Millionen professionelle Entwickler unstützten, reproduzierten und erweiterten die MS-Plattform. (Newman, op.cit.). Zur Pflege des Netzwerks von externen Entwicklern gehören MSs Lizenzen mit rd. 50 Universitäten und staatlichen Forschungslabors, denen Quelltext für Windows-NT u.a. bereitgestellt wird. Auch OEMs und anderer Technologiepartner erhalten Zugang zu Quellcode, nicht jedoch das Recht, abgeleitete Software zu verwerten. MS profitiert vom Austausch mit *Cutting-Edge*-Informatikern. Gelegentlich kommt auch Code an MS zurück, der aber erst der firmeninternen Qualitätskontrolle unterzogen wird. (Cusumano/Selby 1997)

zu einer eigenständigen Ware und zugleich zu einem Massenmarkt. Seither kaufen Privatanwender ebenso wie Firmen ihre Software in *Shrink-Wrapped*-Packungen von der Stange. Im nächsten Schritt, den nun seinerseits MS zu verschlafen schien, eröffnete das Internet ein neues Marktsegment, in dem MS sich neue Gelegenheiten für vertikale Integration und *Bundling* erschloß.

Der Verleger Tim O'Reilly erzählt immer wieder gern die Geschichte von einer computerlosen Freundin, die ihm eines Tages sagte, sie wolle einen Computer kaufen, um Amazon benutzen zu können. Online-Buchbestellung ist eine "Killer-Applikation", die Leute dazu bringt, sich einen Rechner zuzulegen, ähnlich wie es Tabellenkalkulationsprogramme in den frühen achtziger Jahren waren. O'Reilly begann nach diesem Erlebnis, Amazon und Yahoo als Applikationen zu begreifen, nicht als Websites. "A web site was something you 'published' and it was all about documents that were put up on line. But applications are about things that people do."[77] Er bezeichnet sie nicht als Software-Applikationen, sondern als *Infoware Applikationen.* Natürlich beruht auch Infoware auf Soft- und Hardware, doch der Schwerpunkt liegt nicht in der Nutzeroberfläche oder der Funktionalität, sondern in den Inhalten. Das alte Paradigma sei: wenig Information eingebettet in viel Software (in MS Word z.B. ein bißchen Hilfe-Information in einem Pull-Down-Menü). Nach dem neuen ist in vergleichsweise wenig Software (z.B. Perl-Skripten) viel Information 'eingebettet' (z.B. eine riesige Datenbanken, aus der dynamisch Webseiten generiert werden[78]). Die herkömmliche *Bloatware* bietet ihren Kunden immer mehr Menüs für ihr Geld. Eine *Infoware*-Anwendung im Sinne O'Reillys manifestiert sich durch nicht mehr als bspw. einen "What's related"-Knopf im Browser oder in der Suchmaschine. Dahinter stehen jedoch aufwendige semantische Modelle und Verknüpfungen, die beständig erneuert werden. "It's software as a process much more than as a product."[79]

Für das neue Infoware-Paradigma steht Html als einfaches Format mit geringer Einstiegshürde durch einen "View Source"-Button, der es jedem erlaubt, von den Webseiten anderer zu lernen. Man muß kein Programmierer sein, um Webseiten zu bauen. Daher betrachtet O'Reilly Html als eine neue Schicht über der Software, die es einer Flut von Menschen ermöglicht, zu Informationsanbietern zu werden. Denselben Effekt wie die Quelloffenheit von Html haben Skripte in Perl, Python oder Tcl, die sich in der Regel ebenfalls jeder herunterladen, sie studieren, modifizieren und für eigene Zwecke verwenden kann. Beide sind charakteristisch für den Geist der Open Source-Gemeinde. Umgekehrt sei Microsofts ActiveX daran gescheitert, daß es versuchte, das Web in ein Software-Produkt zurückzuverwandeln.[80] Im Internet haben wir heute die Protokolle der unteren Schicht wie

---

[77] Tim O'Reilly, Wizards 7/1999

[78] Bei Yahoo beispielweise sind Skripte im Einsatz, die ständig den unaufhörlichen Strom von Börseninformationen aus den unterschiedlichsten Quellen filtern und in den Yahoo-Börsenticker abbilden, so daß eine Surferin einen Börsenwert anklicken kann und die jeweils neuesten Berichte dazu erhält. Es handelt sich also um eine Applikation, die ununterbrochen und in Echtzeit abläuft. Perl wird gern als das "Klebeband" des Internet bezeichnet: etwas, daß es erlaubt, verschiedene Module, wie Datenbanken, Formulare und Webseiten schnell, flexibel und dynamisch miteinander zu verbinden.

[79] Tim O'Reilly, Wizards 7/1999

[80] Nachdem der Versuch, das Web mit Hilfe von ActiveX in ein geschlossenes Software-Produkt zu verwandeln, von den Anwendern zurückgewiesen worden war, reagierte MS darauf mit Active Server Page (ASP) und Umgebungen, die für die Skripting-Szene sehr viel ansprechender sind. Eine Suchmaschinenanfrage nach *.asp und nach *.pl zeigt, daß sie damit Erfolg haben. Microsoft konkurriert direkt mit den freien Skriptsprachen wie Perl, Python oder Tcl. ASP ist kein Industriestandard, sondern wird ausschließlich von MS

TCP/IP, eine mittlere Schicht wie HTTP und darüber XML-basierte Datenaustauschprotokolle. Kritsch für die Zukunft wird sein, wer dieses letzte Schicht kontrolliert.

Vor dem Hintergrund dieser Entwicklungen der vergangenen zwanzig Jahre ist das Auftauchen der freien Software umso verblüffender. Doch auch unter proprietären Anbietern setzt sich die Erkenntnis wieder durch, daß Software keinen Produktmarkt darstellt, sondern einen Dienstleistungsmarkt. So erzielen Firmen wie IBM und DEC den überwiegenden Teil ihrer Einnahmen heute durch Support-Dienstleistungen. Quelloffene Software bietet die ideale Grundlage für eine maßgeschneiderte Anpassung, Wiederverwendung von Modulen und Weiterentwicklung, ohne das Rad neu erfinden zu müssen. Ohne wirklich selbst akademisch zu sein, beerbt sie die Wissenschaftstradition des freien Austausches.[81] Um dem Phänomen näherzukommen, wird im folgenden die von der PC-Welt getrennte Entwicklungslinie der Software für Großrechner und Workstations geschildert, die sich ab Anfang der Neunziger durch PC-Unixe mit der hier nacherzählten zu überschneiden beginnt.

Betriebssysteme

Betriebssysteme sind eine Mnemotechnik. Im Prinzip könnte ein Programmierer bei jedem Projekt mit der nackten Maschine beginnen und seine eigenen Anweisungen für die Ansteuerung der Datenträger, die Struktur der Dateien oder das Aufleuchten von Pixeln auf dem Schirm schreiben. Genau das geschah auch, bis IBM 1962 einen Satz dieser immer wieder benötigten Routinen zusammenstellte und unter der Bezeichnung OS/360 veröffentlichte -- das erste Betriebssystem war in der Welt. Ein Betriebssystem enthält Hardware-nahe Basisoperationen, auf die jedes Anwendungsprogramm zurückgreift. Es stellt also eine infrastrukturelle Schicht über einer Hardware (z.B. der IBM 360er Serie) dar, auf der ein Programmierer aufsetzt, um eine Statistik- oder Textverarbeitungs-Software zu schreiben. Statt sich um Hardware-Manipulation kümmern zu müssen, kann er sich auf Symbolmanipulation konzentrieren. Da sein eigener Code an zahlreichen Stellen mit der zugrundeliegenden Betriebssystemschicht interagiert, muß er wissen, wie sie funktioniert. “So a proprietary, closed, secret operating system is a contradiction in terms. It goes against the whole point of having an operating system.”[82]

Mit dem Aufkommen eines Massenmarktes für *Personal Computers* trat neben die Programmierer eine neue Gruppe von reinen Anwendern. Die beiden Gruppen haben eine sehr unterschiedliche Sicht auf Computer und Software. Betriebssysteme sind die Arbeitsumgebung der Programmierer, die daher zugänglich, flexibel und offen sein müssen. Auch für Anwender ist Software eine Arbeitsumgebung, doch Gegenstand ihrer Arbeit ist nicht die Software selbst, sondern Texte, Bilder, Musik. Das Betriebssystem läuft für sie

---

unterhalten (ASP-FAQ: http://www.asp-zone.com/aspfaq.asp). ASP wird derzeit außer von MS Internet Information Server und MS Personal Web Server nur von O’Reillys WebSite Pro unterstützt.

[81] Auch heute noch wird Software, die an Universitäten entwickelt wird, der Gemeinschaft der Wissenschaftler frei zugänglich gemacht. Beim DFN-Verein laufen z.B. Projekte im Bereich des verteilten Lehrens und Lernens, bei denen Vorlesungen über das Wissenschaftsnetz an andere Universitätsstandort in Deutschland übertragen werden. Dafür werden die M-Bone-Tools und andere freie Software eingesetzt und weiterentwickelt, und stehen dann den Universitäten und Forschungseinrichtungen für ihre eigenen Belange zur Verfügung. (Paffrath (DFN), Fachgespräch 7/1999)

[82] Stephenson 1999

möglichst unsichtbar im Hintergrund. Es ist die Möglichkeitsbedingung für die Ausführung von Applikationen. Nur wenn etwas schiefgeht, tritt es in den Vordergrund (z.B. wenn der Bildschirm sich in den *"Blue Screen of Death"* von MS-Windows verwandelt).

> "[T]he very nature of operating systems is such that it is senseless for them to be developed and owned by a specific company. It's a thankless job to begin with. Applications create possibilities for millions of credulous users, whereas OSes impose limitations on thousands of grumpy coders, and so OS-makers will forever be on the shit-list of anyone who counts for anything in the high-tech world. Applications get used by people whose big problem is understanding all of their features, whereas OSes get hacked by coders who are annoyed by their limitations."[83]

Unix

Da Unix -- das Betriebssystem und die Kultur seiner Entwickler und Anwender -- im folgenden eine zentrale Rolle spielen wird, sei hier kurz auf die Zufälle und Wendungen seiner Geschichte eingegangen.

Unix war eine Reaktion auf das ab 1964 gemeinsam vom MIT, General Electric und AT&T entwickelte Multics. Nachdem der allumfassende Anspruch des Multics-Projekts gescheitert war, zog sich AT&T 1969 aus der Kooperation zurück. Als Ken Thompson von AT&Ts Bell Laboratories daraufhin einen neuen Versuch startete, setzte er auf seinen Erfahrungen mit Multics auf, aber wählte einen entgegengesetzten Design-Ansatz. Statt alles für alle zu sein, sollte Unix aus kleinen, einfachen, effizienten Werkzeugen bestehen, die miteinander kombiniert werden können, um komplexere Aufgaben zu lösen. Es sollte auf verschiedenen Plattformen einsetzbar sein, da AT&T Rechner verschiedener Hersteller im Einsatz hatte. Und es sollte Time-Sharing unterstützen, also eine interaktive Computer-Nutzung (statt, wie bislang, Stapelverarbeitung), die Anfang der sechziger Jahre erstmals im CTSS (Compatible Time-Sharing System) des MIT implementiert und im Multics-Projekt weiterentwickelt worden war. Dennis Ritchie, der Co-Autor von Unix, schrieb: "What we wanted to preserve was not just a good programming environment in which to do programming, but a system around which a fellowship could form. We knew from experience that the essence of communal computing, as supplied by remote-access, time-shared machines, is not just to type programs into a terminal instead of a keypunch, but to encourage close communication."[84]

Um eine Portierbarkeit auf andere Rechner zu erreichen, wurde Unix 1971 in C umgeschrieben. Damit führte es das Konzept der Source Code-Kompatibilität ein. Der Objektcode eines Programms läßt sich nicht von einem Unix-System auf ein anderes übertragen, doch der Quellcode kann auf dem Zielsystem zu einer ablauffähigen Version kompiliert werden.[85] Im selben Jahr machte AT&T Unix offiziell zum internen Standard-Betriebssystem.

---

[83] Stephenson 1999

[84] zitiert nach Hauben/Hauben, Kapitel 9

[85] "Source-Code-Kompatibilität war eben das Konzept, das Unix auch vorangebracht und am Leben erhalten hat, allen Anstrengungen internationaler Software-Konzerne zum Trotz, das ganze zu Fall zu bringen." (Patrick Hausen, Wizards 7/1999)

Als staatlich reguliertem Telefonmonopol war es AT&T untersagt, sich in anderen Wirtschaftsbereichen zu engagieren. Es war also aus kartellrechtlichen Gründe nicht in der Lage, Unix regulär zu vermarkten. Stattdessen gaben die Bell Labs den Unix-Quellcode gegen Selbstkosten (etwa $ 50, ähnlich den heutigen Linux-Distributionen) an Universitäten ab. Die Kompaktheit, Modularität und Zugänglichkeit durch C ermunterte viele Benutzer, eigene Entwicklungen durchzuführen, so daß Unix schnell einen hohen Reifegrad erreichte. "Dies ist deshalb bemerkenswert, da kein Entwicklungsauftrag hinter diesem Prozeß stand und die starke Verbreitung von Unix nicht auf den Vertrieb oder die Werbung eines Herstellers, sondern primär auf das Benutzerinteresse zurückzuführen ist."[86]

Um die Wartungsarbeiten auf entfernten AT&T-Rechnern zu erleichtern, wurde an den Bell Labs ein Verfahren zur automatischen Herstellung von Telefonverbindungen und Übertragung von Software entwickelt. UUCP (Unix to Unix Copy) wurde als Teil der Unix Version 7 (1978) ausgeliefert. Da AT&T für ihr Unix keinerlei Support anbot, blieb den Nutzern gar nichts anderes übrig, als sich zu einer *Community* zusammenzufinden. Diese kommunizierte bereits mit Hilfe von Newslettern und Konferenzen, doch mit UUCP bekam sie ein Medium innerhalb der Unix-Umgebung selbst zur Verfügung, noch bevor die meisten Universitäten Zugang zum ARPANET hatten. Dateiübertragung, eMail und Netnews, die bereits vorher zur Bildung von lokalen *Communities* um jeweils einen interaktiven, *Multiuser-Time-Sharing*-Rechner geführt hatten, konnten jetzt eine weltweite *Community* potentiell aller Unix-Nutzer unterstützten. Elektronische 'schwarze Bretter' dienten bereits den Dutzenden oder Hunderten von Nutzern einer einzelnen PDP-11 zum Austausch nicht nur über Computer-bezogene Themen. 1979 lud Thompson einen Doktoranten an der Duke Universität und ebenfalls leidenschaftlicher Computer-Schachspieler, Tom Truscott, zu einem Sommer-Job an die Bell Labs ein. Nachdem Truscott aus dem Unix-Himmel an die Duke Universität zurückgekehrt war, begann er im Herbst des Jahres, die Usenet-Software zu schreiben. Durch den *Polling*-Mechanismus von UUCP erlaubt sie einem Rechner, zu einer festgelegten gebührengünstigen Tageszeit einen anderen Rechner anzurufen und die neuesten Nachrichten auszutauschen. Nach einigen solcher Verbindungen waren alle neuen Nachrichten auf allen an das Usenet angeschlossenen Rechnern[87] angekommen. Bald wurde an der Berkeley Universität ein *Gateway* zu den Mailinglisten des ARPANET eingerichtet. In kürzester Zeit hatten Universitäten, Unternehmen (DEC, Microsoft, Intel usw.) und Forschungseinrichtungen Zugang zum Usenet.[88]

In den Newsgroups wurde über Schach, Science Fiction und das allgegenwärtige 'Weltnetz' der Zukunft debattiert, vor allem aber war das Usenet das Medium, in dem die Mitglieder der Unix-*Community* sich gegenseitig unterstützten. Hier baten Nutzer um Hilfe und boten ihre Erfahrungen und ihre Programme an, so daß andere darauf aufbauen konnten, ohne das Rad immer wieder neu erfinden zu müssen.

> "People often look at each other's code, comment on it in person and through interuser communication facilities, and take pieces of it for their own use. The ideas of programming teams and egoless programming fit into the Unix environment well,

---

[86] Gulbin/Obermayr 1995: 7

[87] 1980 waren es acht über die USA verstreute Computer, 1982 waren es schon 50 und die erste transatlantische Verbindung nach Amsterdam stand. (Hauben/Hauben 1996, Chapter 10)

[88] 1980 hatten mehr als 90% aller Informatiklabors in den USA einen oder mehr Unix-Rechner. Auch in Europa und Australien begann sich das Betriebssystem zu verbreiten.

> since they encourage sharing rather than isolation. ... The code that people see, adapt, and imitate is usually well structured. People learn to code well in the same way that they learn to speak their native language well by imitation and immediate feedback."[89]

Die Bell-Forscher um Thompson waren sehr bereitwillig, ihre Wissen in den Newsgroups mit der Unix-*Community* zu teilen. Gelegentlich verbreiteten sie auch Datenbänder mit *Fixes*, Verbesserungen und Zusätzen, doch ihre Hauptaufgabe war es, Unix für den Gebrauch innerhalb von AT&T weiterzuentwickeln. Zum Zentrum der akademischen Forschung dagegen wurde Thompsons Alma Mater, die University of California at Berkeley, die 1974 ihr erstes Unix auf einer neuen PDP-11 installiert hatte. Da es vergleichsweise leicht zu studieren und zu lehren war und noch klein genug, um von einem Einzelnen überschaut zu werden,[90] wurde es unter den Studenten schnell populär. Einige ihrer Programme, wie die Verbesserungen am Pascal-Compiler und der Editor *ex*, waren sehr gefragt, so daß ein Doktorant namens Bill Joy 1977 die erste "Berkeley Software Distribution" (BSD) zusammenstellte, von der im Laufe des Jahres etwa 30 freie Kopien verschickt wurden.[91]

Joy entwickelte *ex* weiter zu *vi*, integrierte die Reaktionen auf das Pascal-System und stellte sie in 2.11BSD (1978) zu einer vollständigen Unix-Distribution zusammen. Berkeley steuerte selbst zahlreiche weitere Innovationen und Ports auf neue Hardware bei und übernahm die Sammlung der Unix-Erweiterungen aus der akademischen Unix-*Community*, während parallel dazu AT&T mit wachsendem Nachdruck auf stabile kommerzielle *Releases* an der eigenen Version von Unix weiterarbeitete.

In dieser Zeit suchte das ARPANET-Projekt nach einer Möglichkeit, die Computer-Umgebung des Netzes zu vereinheitlichen. Von allen Knotenbetreibern zu verlagen, daß sie dieselben Rechner anschafften, war ausgeschlossen, also beschloß man, die Vereinheitlichung auf der Ebene des Betriebssystems vorzunehmen. Durch seine nachgewiesen einfache Portierbarkeit fiel die Wahl auf Unix. 1980 gelang es der Berkeley Universität einen Forschungsauftrag der ARPA (Advanced Research Projects Agency des US-Verteidigungsministeriums) zu erhalten, um BSD nach den Anforderungen der ARPANET-Gemeinde weiterzuentwickeln. Bill Joy wurde zum Projektleiter.[92] Die Anwälte von AT&T und der Universität erarbeiteten eine Lizenz, mit der alle Seiten leben konnten. Zu den wichtigsten Berkeley-Innovationen dieser Periode gehören ein schnelleres Dateisystem, ein Mechanismus für die Kommunikation zwischen Prozessen, der verteilte Systeme möglich machte, und vor allem die Integration der ARPANET-Protokollfamilie TCP/IP in das BSD-Unix. Die neue stabile und dokumentierte Version wurde im August 1983 als 4.2BSD *released*.

> "The popularity of 4.2BSD was impressive; within eighteen months more than 1,000 site licenses had been issued. Thus, more copies of 4.2BSD had been shipped than of all the previous Berkeley software distributions combined. Most of the Unix vendors shipped a 4.2BSD system rather than the commercial System V from AT&T. The

---

[89] Brian Kernighan und John Mashey, zitiert nach Hauben/Hauben 1996, Chapter 9

[90] Version 6 (1975) hatte weniger als 10.000 Zeilen Code.

[91] McKusick 1999: 33 f.

[92] Er verließ Berkeley 1983, um zusammen mit Andreas von Bechtolsheim, Scott McNealy und Vinod Khosla Sun Microsystems zu gründen.

reason was that System V had neither networking nor the Berkeley Fast File System."[93]

Im Jahr darauf wurde AT&T aufgeteilt.[94] Mit dem Verlust des Monopols war es frei, Unix deutlicher als kommerzielles Produkt zu vermarkten, als es das ohnehin schon seit einigen Jahren getan hatte. D.h., es bot Schulungen, Support, Wartung und Dokumentation an, machte Werbung für Unix und veränderte die Lizenzen. Zu diesem Zweck gründete AT&T das Tochterunternehmen Unix System Laboratories (USL).[95] Auch vorher schon hatten andere Hersteller den von AT&T lizenzierten Unix-Code auf bestimmte Hardware-Plattformen oder Anwendungsgebiete hin optimiert und ihre eigenen Unix-Versionen vermarktet. Neben AT&Ts und BSDs Unix entstanden weitere untereinander zunehmend inkompatible Verzweigungen der Code-Basis. AT&T hatte versucht, mit System V.0 (1983) die Unix-Welt auf einen einheitlichen Standard festzulegen, und erneut im System V.4 (1990) die divergierenden Hauptlinien in einem einzigen Unix zu integrieren. Zu den heutigen kommerziellen Unixen, die meist von USLs Unix V.4 ausgehen, gehören AIX (IBM), HP/UX (Hewlett Packard), SCO-Unix (SCO), Sinix (Siemens), Sun/OS, Solaris (Sun), Unixware (Novell), Ultrix und OSF/1 (DEC). Nach dem Konvergenzpunkt von V.4 laufen die Entwicklungslinien aufgrund von lizenztechnischen Mechanismen und der vermeintlichen Marktnotwendigkeit, sich durch proprietäre Zusätze von den Konkurrenten zu unterscheiden,
wieder auseinander. Die Quellcode-Kompatibilität geht verloren. Anwendungshersteller müssen Versionen für die einzelnen Unix-Varianten anbieten.

Microsoft spielt in der Geschichte nur eine Nebenrolle, dafür aber eine besonders unrühmliche. MS hatte in den frühen Achtzigern den Unix-Code von AT&T lizenziert und eine eigene Variante namens Xenix entwickelt, ein von Version 7 ausgehendes, für 16-Bit-Rechner optimiertes Unix. Da es auf PCs lief, war Xenix lange Zeit das Unix mit der größten Zahl von Installationen. Einen Teil des MS-Codes nahm USL zusammen mit Teilen aus dem BSD-basierten SunOS in einer Bemühung, ein vereintes Unix zu schaffen, in sein System V.4 (1990) auf.[96] Als Santa Cruz Operation (SCO) 1995 diesen Code von dann USL/Novell erwarb, um ein Unix für die Intel-Plattform zu entwickeln -- also einen direkten Konkurrenten zu Windows NT --, verklagte MS SCO darauf, daß es nicht nur Lizenzgebühren in Höhe von $ 4 Millionen im Jahr an MS zahlen mußte, sondern gar daß es SCO untersagt sei, neue Versionen von Unix zu entwickeln, die den MS-Code nicht mehr

---

[93] McKusick 1999: 38

[94] Bereits 1912 sah sich die Regierung gezwungen, gegen die Marktmacht des von Alexander Graham Bell gegründeten Unternehmens vorzugehen. Weil die Gesetzgeber den Vorteil eines einheitlichen, flächendeckenden Telefonnetzes sahen, gewährten sie AT&T mit dem Willis-Graham Act von 1921 eine staatlich regulierte Monopolsstellung. Mit der Carterphone-Entscheidung von 1968 und dem Urteil im Kartellverfahren des US-Justizministeriums von 1974 wurde die Auflösung des Monopols vorbereitet, doch dauerte es noch bis 1984, bis AT&T tatsächlich in ein Stammunternehmen (AT&T), das auf dem deregulierten Markt für Ferngespräche konkurrierte, sieben regionale Baby-Bells und die F&E-Einrichtungen (Bellcore) aufgeteilt wurde.

[95] AT&T versuchte vergeblich, der freien Kooperation des Netzes eine geschlossene Industriekooperation entgegenzusetzen, indem es einigen großen Unix-Firmen Minderheitsbeteiligungen an USL anbot, "bis 1993 die Firma Novell USL vollständig übernahm -- sehr zum Ärger vieler Unix-Anbieter, die um die Unabhängigkeit der Unix-Quellen und deren Zugang fürchteten." (Gulbins/Obermayr 1995: 8)

[96] Gulbins/Obermayr 1995: 14

enthalten würden. Nach einer Beschwerde von SCO vor der Europäischen Kommission, die SCO darin zustimmte, daß MS gegen europäisches Wettbewerbsrecht verstoßen habe, verzichtete MS 1997 auf Lizenzzahlungen und die Forderung, daß -- veralteter -- MS-Code in allen zukünftigen Versionen von Unix enthalten sein muß.[97]

In der BSD-Welt führte der wachsende Lizenzdruck im Gegenteil zu einer Befreiung des Codes. Bis 1988 mußten alle Nutzer von BSD, das immer mit sämtlichen Quellen ausgeliefert wurde, gleichzeitig eine AT&T-Quellcodelizenz erwerben, und der Preis für eine solche Lizenz stieg nun kontinuierlich. Händler, die BSD-Code verwenden wollten, um TCP/IP-Produkte für den PC-Markt zu entwickeln, baten Berkeley, die Netzwerkelemente aus BSD separat unter einer freieren Lizenz anzubieten. Da der TCP/IP-Netzwerkcode vollständig an der Berkeley Universität entwickelt worden war, konnte sie ihn zusammen mit den umgebenden Werkzeugen 1989 im Networking Release 1 als Berkeleys ersten frei weitergebbaren Code veröffentlichen. Die BSD-Lizenz erlaubte es, den Code in Quell- und Binärform, modifziert und unmodifiziert ohne Gebühren frei zu verbreiten, solange er den Urheberrechtsvermerk der Berkeley Universität enthielt. Berkeley verkaufte die Datenbänder für $ 1000, doch in kürzester Zeit stand der Code auch auf anonymen ftp-Servern zur Verfügung.

Aufgrund des Erfolgs des Networking Release 1 wollte Keith Bostic von der Berkeley-Gruppe weitere Bestandteile des BSD-Unix freigeben. Seine Kollegen waren skeptisch, da es bedeutet hätte, hunderte von Werkzeugen, die riesige C-Bibliothek und den Kernel auf AT&T-Code zu durchforsten und diesen zu ersetzen.

> “Undeterred, Bostic pioneered the technique of doing a mass net-based development effort. He solicited folks to rewrite the Unix utilities from scratch based solely on their published descriptions. Their only compensation would be to have their name listed among the Berkeley contributors next to the name of the utility that they rewrote. ... [W]ithin 18 months nearly all the important utilities and libraries had been rewritten.”[98]

Mike Karels, Bostic und McKusick verbrachten daraufhin die nächsten Monate damit, auch die Kernel-Dateien von AT&T-Code zu befreien, was ihnen mit Ausnahem von sechs Dateien gelang. Das Ergebnis wurde 1991 als Networking Release 2 unter derselben Lizenz wie Release 1 veröffentlicht. Weitere sechs Monate später hatte Bill Jolitz auch die ausstehenden sechs Kernel-Dateien ersetzt. Das jetzt voll funktionstüchtige freie Betriebssystem, kompiliert für den 386er PC, stellte Jolitz Anfang 1992 unter dem Namen 386/BSD ins Netz.

Das Unix auf dem PC wurde begeistert aufgenommen und bald setzte eine Flut von *Bug-Fixes* und Erweiterungen ein, die Jolitz nicht mehr alleine bewältigen konnte. So bildeten die regesten Nutzer die NetBSD-Gruppe, die das System unterhielt und weiterentwickelte. Auf NetBSD und die Varianten FreeBSD und OpenBSD wird unten unter

---

[97] the Commission agreed with SCO that enforcement of the Microsoft/SCO agreement "in perpetuity" constituted an infringement of the European competition law in that it "impedes technical progress, and in particular hampers [SCO's] ability to compete with Microsoft's own products, particularly Windows NT." N.N., “Microsoft Releases SCO From Obligation to Include, and Pay Royalties on, Outdated UNIX Code”, in: TechMall, Nov. 24, 1997, http://www8.techmall.com/techdocs/TS971124-6.html

[98] McKusick 1999: 42

den freien Software-Projekten näher eingegangen. Zusätzlich zu den freien Gruppen bildet sich auf Grundlage des Networking Release 2 die Firma Berkeley Software Design, Incorporated (BSDI). Auch sie fügte die fehlenden sechs Kernel-Dateien hinzu und verkaufte 1992 ihre kommerziell unterstützte Version einschließlich Quellcode für $ 995. Der Preis für Unix System V von AT&Ts USL inclusive Quellcode lag zu diesem Zeitpunkt bereits bei $ 100.000.

USL strengte daraufhin eine Klage gegen BSDI an, damit diese für ihr Produkt nicht länger den markenrechtlich geschützten Namen “Unix” verwendeten und da es proprietären USL-Code und Handelsgeheimisse enthalte. Nach einem zweijährigen Prozeß, in den sich auch die Berkeley Universität mit einer Gegenklage einschaltete und in dessen Verlauf USL von Novell aufgekauft wurde, mußten schließlich drei der 18.000 Dateien aus dem Networking Release 2 entfernt und einige andere geringfügig geändert werden.[99]

Die derart gesegnete Version wurde 1994 unter dem Namen 4.4BSD-Lite veröffentlicht. Sie wurde weiter gepflegt, bis die Bug Reports und Erweiterungen versiegten. Die letzten Änderungen wurden im Juni 1995 als 4.4BSD-Lite, Release 2 vorgelegt. Die Forschungsgruppe an der Berkeley Universität wurde aufgelöst. Nachdem sie fast zwanzig Jahre lang die freie akademische Entwicklung von Unix getragen hatte, war es Zeit für andere, die Stafette zu übernehmen.

Das GNU-Projekt

Als Richard Stallman 1971 seine akademische Laufbahn am KI-Labor des MIT aufnahm, gab es noch keine unfreie Software, nur autoritärere und freiere Informatikinstitute. Harvard, wo Stallman als Experte für Assemblersprachen, Betriebssysteme und Texteditoren gearbeitet hatte, gehörte zur ersten Kategorie. Auf der Suche nach einer Hacker-freundlicheren Atmosphäre wechselte er dann als Systemprogrammierer ans MIT, dessen Labor für Künstliche Intelligenz ein damals bereits legendäres Hacker-Paradies war, ein Kloster, in dem man lebte, um zu hacken und hackte, um zu leben. Steven Levys 1984 geschriebener Klassiker “Hackers. Heroes of the Computer Revolution”[100] verfolgt das Phänomen zurück bis in den Modelleisenbahnclub am MIT der späten Fünfziger.[101] Was Stallman am KI-Lab

---

[99] McKusick 1999: 44 f.

[100] Levy 1994 (Die Ausgabe von 1994 enthält ein neues Nachwort)

[101] Im Club gab es zwei Fraktionen, eine, die es liebte, Modellhäuser, Landschaften und Replikas historischer Züge zu bauen -- heute würde man sie die Interface-Designer nennen. Die andere Fraktion verbrachte die meiste Zeit mit dem Steuerungssystem unter der Platte, mit der Stromversorgung, den Kabeln und elektromagnetischen Relais, die sie von einem Telefonhersteller bekommen hatten. Diese zweite Gruppe strebte nach Höherem, doch der zentrale MIT-Rechner, eine IBM 704, die Lochkartenstapel verarbeitete, war von der Computerpriesterschaft abgeschirmt. Als das MIT 1959 einen der ersten Transistor-betriebenen Rechner der Welt bekam, der außerdem mit einem Kathodenstrahlmonitor ausgestattet war, verloren sie bald das Interesse an Modelleisenbahnen. Die TX-0 des Lincoln Labs war ein Zwerg im Vergleich zur 704, verwendete keine Lochkarten mehr und man bekam Zeitabschnitte zugewiesen, in denen man ihn exklusiv für sich benutzen konnte. Zum ersten Mal konnte man am Computer sitzen, während er ein Programm durchrechnete, und auf der Stelle neue Anweisungen in die Tastatur hacken. Während bislang und auch später beim ‘strukturierten Programmieren’ der größte Teil des Software-Entwurfs abstrakt auf Papier stattfand, war es mit der neuen ‘interaktiven’ Computer-Nutzung möglich, eine Idee in die Tasten zu hacken, das Programm laufen zu lassen, Fehler zu entdecken, die Korrekturen einzugeben und es sofort wieder ablaufen zu lassen. Diese Art der iterativen ad-hoc-Programmierung trug den Namen ‘Hacken’. (Levy 1994: 21 ff.)

mochte, war, “there were no artificial obstacles, things that are insisted upon that make it harder for people to get any work done -- things like bureaucracy, security, refusals to share with other people.”[102]

Dort traf Stallman auf Hacker-Legenden wie Richard Greenblatt und Bill Gosper und tauchte in eine Kultur des freien Wissensaustausches ein, eine Oase der konstruktiven Kooperation im allgemeinen *dog-eat-dog*-Dschungel.

> “I had the good fortune in the 1970s to be a part of a community in which people shared software. We were developing software, and whenever somebody wrote an interesting program it would circulate around. You could run the program, add features, or just read the code and see how problems were solved. If you added features to the program then other people could use the improved version. So one person after another would work to improve the software and develop it further. You could always expect at least the passive cooperation of everybody else in this community. They might not be willing to drop their work and spend hours doing something for you, but whatever they had already done, if you could get some use out it, you were welcome to do so.”[103]

Neben seiner Arbeit als Systementwickler und an Emacs erwarb er gleichzeitig einen magna-Abschluß in Physik an der Harvard Universität. Emacs, das 'Schweizermesser' unter den Editoren, war damals Stallmans bekanntestes Werk. Basierend auf einem Lisp-Dialekt, ist es beliebig konfiguierbar und erweiterbar. Seine weit offene Architektur ermunterte viele, Zusätze und Verbesserungen zu schreiben. Stallman betrieb das Projekt Emacs im selben *sharing spirit*, den er am KI-Lab schätzte. Er gab das Programm frei an jeden weiter, unter der Bedingung, daß alle, die Erweiterungen schrieben, diese mit der Emacs-*Community* teilten.

In den ausgehenden Siebzigern und frühen Achtzigern erlebte Stallman jedoch auch den Verfall der Hacker-Ethik und die Auslöschung der Gemeinde am KI-Lab mit. Es begann damit, daß auf den Systemen des MIT-Rechenzentrums Passwörter eingeführt wurden. Als echter Hacker verachtete Stallman Passwörter. Die Rechner, die er am KI-Lab administrierte, hielt er frei davon und auf den anderen führte er einen Feldzug zur Durchsetzung des *empty string passwords* (ein schlichtes Return), bis schließlich das US-Verteidigungsministerium drohte, das KI-Lab vom ARPAnet abzuhängen. In einem zunehmend wichtigeren Netz ging es nicht an, daß jedermann, ohne eine Legimation vorzuweisen, durch diese weit offene Tür spazieren und sich in militärischen Rechnern tummeln konnte. Stallman und andere waren der Ansicht, genau so sollte es sein. Doch der Schließungsdruck wurde stärker, und nach und nach verließen die Hacker der ersten und zweiten Generation das MIT, um für Computerfirmen zu arbeiten, eigene zu gründen oder gar zu heiraten.

Die nachwachsende Generation, die Stallman jetzt als ‘Touristen’ auf seinen Lab-Rechnern beobachtete, war nicht in die Hacker-Ethik hineingewachsen. Nicht alle sahen offene Systeme als Chance, um Gutes zu tun und zu lernen, um selbst einmal den Titel eines ‘echten Hackers’ zu erwerben. Viele von ihnen sahen auch nichts Verkehrtes in der Idee eines Eigentums an Programmen. Wie ihre Vorgänger schrieben sie aufregende Software, doch immer häufiger tauchte beim Starten ein Copyright-Vermerk auf dem Schirm auf.

---

[102] Levy 1994: 416

[103] Stallman, Wizards 7/1999

Gegen diese Form der Schließung des freien Austauschs kämpft Stallman bis heute. “I don’t belive that software should be owned,” zitiert Steven Levy ihn im Jahr 1983, “Because the practice sabotages humanity as a whole. It prevents people from getting the maximum benefit out of the program’s existence.”[104]

Die zunehmende Kommerzialisierung war auch der Grund für das Ende der Hacker-Kultur am KI-Lab. Seit 1975 hatte Richard Greenblatt zusammen mit einigen anderen an einem Hacker-Traum, der LISP-Maschine, gearbeitet, einem Computer, dessen Architektur speziell für die Anforderungen dieser mächtigsten und flexibelsten, aber auch ressourcenhungrigen Programmiersprache zugeschnitten war. Nach jahrelanger Entwicklungsarbeit hatten die MIT KI’ler schließlich 32 LISP-Maschinen gebaut. Greenblatt begann seine *cutting-edge* Technologie als Element in Amerikas Kampf mit Japan um die Führung in der Künstlichen Intelligenz zu sehen. Und dieses Potential sah er am besten durch ihrer Verbreitung durch den kommerziellen Sektor verwirklicht. Greenblatt wollte eine Firma gründen. Aus den Auseinandersetzung mit seinen Hacker-Kollegen am MIT und externen Beratern ging erst LISP Machine Incorporated (LMI) und ein knappes Jahr später die hochkapitalisierte Firma Symbolics hervor. Die beiden Firmen warben fast alle verbliebenen Hacker vom KI-Lab ab. An die Stelle des *sharing spirit* war eine Konkurrenz um Marktanteile für dasselbe Produkt getreten, mit allen kommunikationswidrigen Umständen davon, wie Vertraulichkeitsvereinbarungen (*Nondisclosure Agreement*, NDA).

> “[T]hey could not talk about what mattered most -- the magic they had discovered and forged inside the computer systems. The magic was now a trade secret, not for examination by competing firms. By working for companies, the members of the purist hacker society had discarded the key element in the Hacker Ethic: the free flow of information.”[105]

Als “last survivor of a dead culture”[106], wie er sich selbst bezeichnete, blieb Richard Stallman zurück. Das lebende Beispiel dafür, daß eine ‘anarchistische und großartige Einrichtung’ möglich ist, war ausgelöscht.

> “If I told people it’s possible to have no security on a computer without people deleting your files all the time, and no bosses stopping you from doing things, at least I could point to the AI lab and say, ‘Look, we are doing it. Come use our machine! See!’ I can’t do that any more. Without this example, nobody will believe me. For a while we were setting an example for the rest of the world. Now that this is gone, where am I going to begin from?”[107]

Steven Levy endet sein 1984 geschriebenes “Hackers” auf einer positiven Note. Das Hacker-Zentrum am MIT war verschwunden, doch sein Geist -- so Levys Fazit -- hat sich mit dem

---

[104] Levy 1994: 419; s.a. Stallman 1994

[105] Levy 1994: 424

[106] ebd.: 427. Stallman führte daraufhin einen Rachefeldzug gegen den vermeintlichen Schurken Symbolics, indem er die neueste Software, die er für MITs Symbolics-Maschine bekam, reverse engineerte und die Ergebnisse an LMI weitergab. Auf diese Weise hackte er gegen mehr als ein Dutzend Weltklasse-Hacker an. Greenblatt bemerkte bewundernd: “[H]e’s been out-hacking the whole bunch of them.” (ebd.:426)

[107] ebd.: 427

persönlichen Computer allgemein verbreitet. Millionen von Menschen wurden der Magie ausgesetzt. Die Hacker-Ethik war vielleicht nicht mehr so rein wie in den Jahren der Priesterschaft, und tatsächlich hatten die Hacker der dritten Generation ihre eigenen Vorstellungen, doch weiterhin bietet jeder PC die Chance, die Magie zu entdecken, die Kreativität anzustacheln und -- ein Hacker zu werden.

Doch bevor sich Levys Prognose machtvoll beweisen sollte, trat die Computerwelt in eine dunkle Phase ein. Anfang der Achtziger war fast alle Software proprietär. Den Begriff 'Hacker' hatte die Presse inzwischen zu 'Computer-Einbrecher' verkehrt. Seit 1981 kann in den USA Software, die bis dato als Algorithmen oder mathematische Formeln und damit als unschützbar angesehen wurde, zum Patent angemeldet werden. DEC stellte seine PDP-10-Serie ein, für die die KI-Lab-Hacker das freie *Incompatible Timesharing System* (ITS) geschrieben hatten -- unter den Betriebssystemen der bevorzugte Tummelplatz für Hacker. Die neue Generation von Rechnern, wie die VAX oder der 68020, kamen mit ihren eigenen proprietären Betriebssystemen. 1984 wurde AT&T aufgeteilt. Die daraus hervorgegangene Software-Firma sah sich jetzt in der Lage, Unix zu vermarkten, und sie tat das in Form von Binärcode und gegen erhebliche Gebühren. Diese Privatisierung und Schließung einer Software, an der zahllose Leute in der ganzen Welt mitgearbeitet hatten, brachte viele von ihnen auf.

Als Schlüsselerlebnis nennt Stallman eine Episode Anfang der Achtziger um einen Netzwerkdrucker am MIT. Es kam regelmäßig vor, daß jemand einen Druckauftrag abschickte und einige Zeit später in den Raum ging, wo der Xerox-Drucker stand, nur um festzustellen, daß sich das Papier gestaut hatte oder ausgegangen war. Stallman wollte nun ein Funktion zufügen, die den Druckerstatus direkt am Arbeitsplatz ausgab. Er fand auch jemanden bei Xerox, der den Quellcode des Druckertreibers hatte, doch weigerte dieser sich, ihn herauszugeben, da er sich zu einer Nichtweitergabe verpflichtet hatte.[108] In dieser Erfahrung verdichtete sich der neue Geist der Zeit: Ein praktisches Problem stellt sich. Die Lösung besteht darin, eine bestehende Software um eine Funktion zu erweitern. Früher hätte man den Autor der Software um den Quellcode gebeten und hätte diesen fortgeschrieben -- die Technologie wäre für alle Betroffenen nützlicher geworden. Doch jetzt stand vor dem Quellcode und vor einer Kooperation von Programmieren eine Mauer namens *intellectual property*. Eine Lösung war damit nicht unmöglich geworden, doch die Firma zu bitten, das *Feature* zu implementieren und es im nächsten *Update* zu verbreiten, ist langwierig und unsicher, und ein *reverse engineering* ist mühsam, zeitraubend und nur in engen Grenzen legal.

Stallman fragte sich also, was er tun könne, um erneut die Voraussetzungen für eine Gemeinschaft zu schaffen. Um einen Computer zu betreiben, benötigt man allererst ein Betriebssystem. Betriebssysteme waren eines von Stallmans Spezialgebieten. Also startete er 1984 das GNU-Projekt. Das rekursive Akronym steht für "GNU's not Unix", doch genau das war sein Ziel: ein Betriebssystem zu schreiben, das funktional äquivalent zu Unix ist, aber keine einzige Zeile von AT&T geschützten Code enthält und vor allem: das in freier Kooperation weiterentwickelt werden kann, ohne irgendwann dasselbe Schicksal zu erleiden wie Unix. Die Wahl fiel auf Unix und nicht ein anderes Betriebssystem, weil es sich bewährt hatte, weil es portabel ist und weil es bereits eine aktive weltweite Unix-Gemeinde gab, die durch seine Kompatibilität leicht zu GNU wechseln konnten.

---

[108] z.B. in Interview mit Richard Stallman von Robin Hohmann, in Computerwoche, Nr. 4 vom 29. Januar 1999, http://www.computerwoche.de/info-point/heftvorschau/detail.cfm?SID=5556

Stallman kündigte seinen Job am MIT, weil seine Arbeit als Angestellter (*work for hire*) der Universität gehören würde, die damit die Vertriebsbedingungen seiner Software bestimmen konnte. Er wollte verhindern, daß er erneut eine Menge Arbeit investierte, nur um hilflos zuzusehen, wie das MIT ein proprietäres Software-Paket daraus machte. Er rechnet es dem damaligen Leiter des KI-Labs hoch an, daß er ihn einlud, auch nach seiner Kündigung die Einrichtungen des Labors zu benutzen.[109]

Im September 1983 kündigte er in Unix-Newsgroups sein Projekt einer "neuen Unix-Implementation" an und lud zur Mitarbeit ein.[110] Er startete, zunächst noch allein, mit dem GNU C-Compiler (GCC) und seinem Editor GNU Emacs.

> "So we started writing the components of this system. The design of this system is that there are many separate programs that communicate with each other; and it was documented, so that you could understand what the interface was between these parts, write each part one by one, and then finally test each part, as a replacement for that part of a Unix system. When you have all the parts replaced, then you put the replacements together and have an entire system. And that is how we did it. It was a very decentralized way of doing things which was well suited to a rather amorphous and decentralized community of volunteers around the world communicating mainly by email."[111]

Sein ursprüngliches Emacs für die PDP-10 hatte er auf einem anonymous ftp-Server verfügbar gemacht und alternativ Interessierten angeboten, ihm einen frankierten Rückumschlag und ein leeres Datenband zu schicken, auf das er dann die Software spielten. Da er kein Einkommen mehr hatte, bot er -- neben der weiterhin kostenlosen ftp-Distribution -- jetzt ein Band mit Emacs für $ 150 an. Als das Interesse an Emacs wuchs, wurde es notwendig, Finanzierungsmöglichkeiten zu erschließen. Zu diesem Zweck wurde 1985 die gemeinnützige Free Software Foundation (FSF) errichtet. Die FSF übernahm die Distribution der Datenbänder, erst für Emacs, dann auch für andere GNU-Software. Die Mittel aus dem Verkauf von Software (in Quellcode und vorkompiliert für bestimmte Plattformen) und Handbüchern, sowie Geld- und Sachspenden verwendet die FSF, um Entwickler dafür zu bezahlen, daß sie bestimmte, für eine vollständige Betriebssystemsumgebung notwendige Programme schreiben.

Im *GNU Manifesto*[112] -- ebenfalls von 1985 -- begründet Stallman die Philosophie der freien Software auf dem Kantschen Imperativ (aka *the Golden Rule*): "I consider that the golden rule requires that if I like a program I must share it with other people who like it."[113] Wer umgekehrt die Nutzungmöglichkeiten eines Programms einschränkt, um Geld von den Nutzern zu extrahieren, verwende destruktive Mittel. Wenn jeder sich so verhalten würde, würden wir all durch die wechselseitige Destruktion ärmer werden. Der Wunsch, für seine Kreativität belohnt zu werden, rechtfertige es nicht, der Welt die Ergebnisse dieser Kreativität vorzuenthalten. Stallman nennt bereits eine Reihe Möglichkeiten, wie Software-

---

[109] Stallman 1999): 57; s.a. http://www.gnu.org/gnu/thegnuproject.html; die Domain gnu.ai.mit.edu ist bis heute in Gebrauch.

[110] das Posting ist archiviert unter http://www.gnu.org/gnu/initial-announcement.html

[111] Stallman, Wizards 7/1999

[112] Stallman 1985

[113] Stallman 1985

Entwickler und -Firmen mit freier Software Geld verdienen können, die in den Achtzigern erprobt werden und in den Neunzigern Schule machen: Vertrieb und Dokumentation; Support in Form von echter Programmierarbeit und 'Händchenhalten' für unerfahrenere Nutzer; Hardware-Hersteller, die für die Portierung eines Betriebssystems auf ihre neue Maschine bezahlen; Schulung; eine User's Group, die gemeinsam einen Programmierer beauftragt, gewünschte Zusätze zu schreiben.[114]

Der wachsenden Tendenz, Information zu horten, hält er einen Begriff von Freundschaft, Gastfreundschaft (*hospitality*) und Nachbarschaftlichkeit entgegen, wie etwas auf den *Wizards of OS*:

> "Another thing you should be able to do [with a program] is make a copy for your friend so that your friend can get the benefit of it too. This is not only useful, this act of cooperation is a fundamental act of friendship among people who use computers. The fundamental act of friendship among beings who can think is to teach each other, to share knowledge. Sharing software is a special case of that, for those of us who use computers. Each act of sharing a copy of a program is not only a useful act, but it helps to reinforce the bonds of good will that are the basis of society and distinguish society from a jungle.
>
> This good will, the willingness to help out your neighbor whenever it's not too hard, is what makes society function and what makes it a decent place to live in. Any kind of policy or any legal system that condemns or prohibits this kind of cooperation is polluting society's most important resource. It is not a material resource, but it is an extremely important resource. "[115]

Im *Manifesto* ist von der Solidarität unter Programmieren die Rede, die sich als Genossen begegnen. "The fundamental act of friendship among programmers is the sharing of programs." Es finden sich auch naturrechtliche Argumentationen: "Copying all or parts of a program is as natural to a programmer as breathing, and as productive. It ought to be as free." Fluchtpunkt der Vision ist eine Welt jenseits des Mangels. Bereits heute sei die Menge der notwendigen Arbeit für die Produktivität der Gesellschaft stark zurückgegangen. Daß sich dies nicht in eine größere Freizeit übersetzt, liege vor allem an den nicht-produktiven Aktivitäten wie Verwaltung und Konkurrenz. "Free software will greatly reduce these drains in the area of software production. We must do this, in order for technical gains in productivity to translate into less work for us."

Im Kern der Vision steht ein freies Software-Universum. "The ultimate goal is to provide free software to do all of the jobs computer users want to do -- and thus make proprietary software obsolete."[116] Das GNU-Projekt ist mehr als nur ein Sammelbecken für diverse freie Programme. Es wird von einer Gesamtvision für ein vollständiges System geleitet. Systematisch wurden alle Bestandteile einer freien Betriebsumgebung in eine *Task*

---

[114] die damalige Idee einer 'Software-Steuer', die von einer Institution wie der NSF benutzt wird, um Software-Entwicklung zu finanzieren, hat sich bald darauf verflüchtigt. Ein privates Umverteilungssystem, bei dem z.B. die FSF Spenden oder Steuermittel entgegennimmt, um Entwickler damit zu bezahlen, ist dagegen durchaus üblich.

[115] Stallman, Wizards 7/1999

[116] Overview of the GNU Project, updated 2 May 1999, jonas, http://www.gnu.org/gnu/gnu-history.html

List eingetragen und nach und nach erarbeitet. Dazu gehören die Befehls-Shell (BASH), Assembler, Compiler (GCC), Interpreter, Debugger (GDB), Editor (Emacs), Archivierer (GNU tar), GNU Make, Mailer und Postscript-Viewer (GNU GhostScript). Bestehende freie Software wie Donald Knuths Textformatierer TeX und das ebenfalls am MIT entwickelte Window-System X Window wurden in GNU integriert.

Zentrales Instrument zur Absicherung dieses expandierenden Universums der freien Software ist die Lizenz, unter der es steht. Die Freiheit, die die GNU Public License (GPL) den Nutzern einer Software gewährt, umfaßt (1) den Zugang zum Quellcode, (2) die Freiheit, die Software zu kopieren und weiterzugeben, (3) die Freiheit, das Programm zu ändern und (4) die Freiheit, das veränderte Programm -- unter denselben Bedingungen -- zu verbreiten. Die Auflage in der 4. Freiheit verhindert, daß freie Software privatisiert und ihrer Freiheiten entkleidet wird. Die GNU-Philosophie schreibt nicht vor, daß die Weiterverbreitung kostenlos zu geschehen hat. Für Dienstleistungen wie Zusammenstellung, Produktion und Vertrieb von CD-ROMs, Support, Schulung und Handbücher ist die Erhebung einer Gebühr ausdrücklich zugestanden, nicht jedoch für die Software selbst. Auf diese vielleicht wichtigste Innovation Richard Stallmanns, die GPL, komme ich im Abschnitt zu Lizenzen zu sprechen.

Mit der GPL von 1989 beginnen die FSF und ihre Advokaten, wie der New Yorker Rechtsprofessor Eben Moglen,[117] auf dem Feld von Copyright und Vertragsrecht zu intervenieren. In anderer Form tut dies auch die *League for Software Freedom*,[118] die 1989 anläßlich Apples *Look-and-Feel*-Verfahren gegen Microsoft von John Gilmore und Richard Stallman gegründet wurde. Die League engagiert sich bis heute gegen User-Interface-Copyrights und Software-Patente.

Mit der Zeit konnten immer mehr zu ersetzende Unix-Komponenten von der *Task List* gestrichen werden, und der Schwerpunkt verlagerte sich auf Anwender-Software, wie ein Spreadsheet, den Grafik-Desktop GNOME, die Kryptographie-Software GNU Privacy Guard und selbst Spiele. Viele der GNU-Komponenten entwickelten ein Eigenleben. Da sie die entsprechenden Komponenten von Unix-Systemen ersetzen konnten und nicht selten besser waren als ihre proprietären Gegenstücke, verbreiteten viele sich als Standardwerkzeuge auch unter Verwaltern kommerzieller Unix-Systeme.

1990 war das GNU-System nahzu vollständig. Die einzige wesentliche Komponente, die noch fehlte, war ein Kernel. Hier war die Wahl auf einen Mikrokernansatz gefallen, Mach, von der Carnegie Mellon University entwickelt und dann von der University of Utah übernommen, sollte als Basis dienen. Darüber sollen die verschiedenen Betriebssystemsfunktionen als eine Sammlung von Servern (*a herd of gnus*) mit dem Namen HURD implementiert werden, doch das Debugging von *multi-threaded* Servern stellte sich als schwieriger heraus, als erwartet.

1991 kam Linus Torvalds dem GNU-Projekt mit dem Linux-Kernel zuvor. Ob nun Linux als letzter Baustein in die GNU-Umgebung eingefügt wurde oder die GNU-Module um Torvalds Kernel -- wie man es auch sehen mag, auf jeden Fall gibt es seither ein vollständiges, leistungsfähiges, freies System mit dem Namen GNU/Linux.

GNU ist das erste 'bewußte' freie Software-Projekt. Die Freiheiten, die in der vorangegangenen Hacker-Kultur unausgesprochene Selbstverständlichkeit waren, wurden nun expliziert und in einem Vertrag zwischen Autoren und Nutzern rechtsverbindlich

---

[117] Moglens Hompage: http://emoglen.law.columbia.edu/

[118] http://lpf.ai.mit.edu/

festgeschrieben. Mit seinen Tools ist GNU außerdem Geburtshelfer aller folgenden Projekte. Schließlich eröffnet sie einen Blick weit über die Software hinaus: “The core of the GNU project is the idea of free software as a social, ethical, political issue: what kind of society do we want to live in?”[119]

GNU/Linux[120]

Ein Zwischenspiel stellt das 1987 von Andrew Tanenbaum entwickelte **Minix** dar, ein Unix-Derivat für 286er PCs, das der Informatikprofessor an der Universität Amsterdam speziell für Lehrzwecke entworfen hat.[121] In der Usenet-Newsgroup comp.os.minix konnten Interessierte seine Entwicklung mitverfolgen. Ende der achtziger Jahre gab es bereits mehr als 50.000 Minix-Anwender.

Darunter befand sich auch der finnische Informatikstudent Linus Torvalds. Er hatte sich einen PC mit dem gerade neu erschienen 386er-Prozessor angeschafft, der erstmals echten Speicherschutz und Multitasking anbot. Ausgehend von Minix begann er, einen neuen 32 Bit breiten, Unix-artigen Betriebssystemskern zu schreiben, der die neuen Möglichkeiten unterstütze, zuerst in Assembler, dann in C. Auch Torvalds veröffentlichte den Quelltext seiner Arbeit im Netz, gab die Adresse in der Minix-Newsgroup kund und lud zur Mitarbeit ein. Die Werkzeuge aus dem GNU-Projekt (C-Compiler, Linker, Editoren usw.) taten dem Projekt gute Dienste. Linux ist heute das Paradebeispiel einer solchen unwahrscheinlichen Organistationsform, bei der Tausende von Menschen in der ganzen Welt in einer verteilten, offenen, locker gekoppelten Zusammenarbeit ein komplexes Software-Projekt entwickeln. “In fact, I think Linus’s cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model.”[122]

Im Januar 1992 lag der bereits stabile Linux-Kernel 0.12 vor, dazu gab es die BASH-Shell, den GNU C-Compiler, eine Emacs-Version und viele weitere GNU-Utilities. Ebenfalls vom GNU-Projekt übernommen wurde die Copyleft-Lizenz, die GNU Public License (GPL), die die Freiheit von Linux und aller abgeleiteter Software sichert. Als Basis einer graphischen Benutzeroberfläche wurde von einem benachbarten freien Projekt XFree86 übernommen. Im März 1994 erschien GNU/Linux Version 1.0. Um eine Aufspaltung des Projekts in verschiedene ‘Geschmacksrichtungen’ (das sog. *Code-Forking*) zu verhindern, etablierte sich unter der weithin akzeptierten Autorität von Torvalds ein System, wonach bestimmte Entwickler für Teilbereiche des Kerns zuständig sind, aber er das letzte Wort darüber hat, was in den Kern aufgenommen wird. Den Projektleitern (*Maintainern*) arbeitet ein große Zahl von Leuten zu, die die Software testen, Bugs melden und beheben und weitere Funktionen hinzufügen. Neben einer Kerngruppe für den Kernel[123] bildeten sich weitere Standardisierungsgremien: die *Linux File System Standard Group* (im August 1993

[119] Stallman, Wizards 7/1999

[120] s. http://www.linux.org/; Linux Newbie FAQ, http://www.tomix.de/linux/faq/

[121] s. Minix Information Sheet, http://www.cs.vu.nl/~ast/minix.html

[122] Raymond 1998

[123] The Linux Kernel Archives: http://www.kernel.org/

gegründet[124]), das *Linux Documentation Project* (LDP)[125] und -- zur Sicherstellung der Kompatibilität der Betriebsumgebung in verschiedenen Distributionen -- die *Linux Standard Base* (LSB).[126]

Die Distribution erfolgte zunächst nur übers Netz, spätestens ab 1993 auch auf Disketten und kurz darauf auf CD-ROM. Ab 1993 stellten kommerzielle Anbieter wie SuSE, Caldera und Yggdrasil Distributionen mit dem neuesten Kernel, Tools und Anwendungen zusammen. Ab 1994 erschienen spezialisierte Zeitschriften (wie Linux Journal, Linux Gazette, Linux Focus und das deutschsprachige Linux Magazin) und Online-Foren (wie Linuxtoday und Slashdot.org). Ab 1993 (und inzwischen in der 7. Auflage von 1997) erscheint das Linux Anwenderhandbuch unter der GPL, das schnell zum Standardwerk im deutschsprachigen Raum wurde.[127] 1995 gab es bereits 500.000 Linux-Nutzer. Ein Jahr darauf waren es 1,5 Mio., 1997 schon 3,5 Mio. und 1998 7,5 Mio. Was als Betriebssystemkern begann und sich um andere freie Software vor allem aus dem GNU-Projekt anreicherte,[128] stellte inzwischen eine Plattform dar, die auch für die Anbieter von Anwendungs-Software interessant wurde. Mehr als 1000 Anwendungen stehen heute auf GNU/Linux zur Verfügung, der größte Teil davon kostenlos.

Auch kommerzielle Office-Pakete von Applixware und Star Division, Corels WordPerfect und Datenbanken von Adabas und Oracle wurden auf Linux portiert. Linuxer können heute zwischen den graphischen Benutzeroberflächen KDE (K Desktop Environment) und Gnome (GNU Network Object Model Environment) mit jeweils verschiedenen Window-Managern wählen. Die Nutzerfreundlichkeit nimmt zu. Während für alte Unix-Admins der normale Weg die Source Code-Installation ist, wird unter Linux Software zunehmend als *Binary* installiert, wie das in der PC-Welt üblich ist. Dazu dienen Installierer wie der RedHat Package-Manager oder dselect. Emulatoren wie Wabi und Wine erlauben es, MS-Windows-Software unter Linux zu nutzen. Mit Werkzeugen wie Wind/U (Bristol Technology) können Entwickler Windows-Quelltexte z.B. in Visual-C++ zu Linux-Anwendungen übersetzen.

Auch die häufig zu hörende Kritik, für Linux gebe es keinen Support, ist gründlich widerlegt. Tausende von engagierten Linux-Nutzern gewähren Anfängern und Hilfesuchenden in Newsgroups und Mailinglisten Unterstützung und reagieren auf Bugs manchmal schon innerhalb von Stunden mit *Patches*. Schon 1997 verlieh Infoworld der Linux-Gemeinde für diesen einzigartig leistungsfähigen Support den *Best Technical Support Award*. Daneben etablierte sich eine wachsende Zahl von Systemhäusern, die Installation, Schulung und Wartung von Linux-Systemen kommerziell anbieten.

Heute gibt es schätzungsweise 20 Mio Installationen weltweit. Linux hat inzwischen auch in Wirtschafts- und Verwaltungskreisen Popularität als zuverlässige und kostengünstige Alternative zu MS-NT vor allem im Server-Bereich erlangt. Zu den Linux-Großanwender gehören Unternehmen wie Edeka, Sixt, Debis und Ikea. Große Computer-Vertreiber begannen ab Mitte 1999, Desktop-Rechner, Workstations und vor allem Server mit

---

[124] vgl. Garrett D'Amore, The Linux File System Standard, http://www2.linuxjournal.com/lj-issues/issue15/1104.html

[125] http://www.linuxdoc.org/

[126] http://www.linuxbase.org/

[127] Hetze/Hohndel/Müller/Kirch 1997

[128] Der Gesamtanteil von GNU-Software an einer Linux-Distribution liegt heute bei ungefähr 30 Prozent.

vorinstalliertem GNU/Linux auszuliefern. Nach der Hochburg Europa und den USA erobert Linux auch den pazifisch-asiatischen Raum von Japan über China bis Indien.[129] Linus Torvalds ist der unangefochtene Star der freien Software. GNU/Linux gelangte durch den Film *Titanic* zu Hollywood-Ruhm, dessen Computergraphiken auf einem Linux-Cluster gerechnet wurden. Seit der LinuxWorld Expo im März 1999 -- eine Art Volljährigkeitsparty für das Projekt -- hat Linux auch seine eigene Industriemesse. Sogar in der Kunstwelt fand es Anerkennung, als dem Projekt der Prix Ars Electronica 1999 (in der Kategorie ".net") verliehen wurde.

Somit nahmen sowohl das GNU-Projekt wie das Linux-Kernel-Projekt ihren Ausgang in Universitäten. Wie Friedrich Kittler betont, war es eine praktische Kritik an der Schließung des wissenschaftlich frei zirkulierenden Wissens, "... daß am Massachusetts Institute of Technology einige Programmierer der Käuflichkeit widerstanden und daß an der Universität Helsinki ein einsamer Informatikstudent die herbeigeredete Angst vor Assemblern und Kaltstarts -- denn darum geht es -- überwand. So direkt waren offene Quellen und freie Betriebssysteme an die Freiheit der Universität gekoppelt. Schauen Sie wieviel 'edu' da drinsteht in den Linux-Kernel-Sources. So direkt hängt aber auch die Zukunft der Universität von diesen freien Quellen ab."[130]

Von 'Free Software' zu 'Open Source Software'

Der Schlüsseltext für die begriffliche Wende in der Debatte ist Eric S. Raymonds "The Cathedral and the Bazaar".[131] In den Versionen bis zum Februar 1998 sprach Raymond darin von 'free Software', dann ersetzte er den Ausdruck durch 'Open Source Software'.[132] *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. Jedenfalls war es erklärtes Ziel der Wende, den mindestens 14 Jahren zuvor von von Richard Stallman geprägten Begriff 'Free Software', unter dem sich die Bewegung weit über das GNU-Projekt hinaus gesammelt hatte, durch einen Begriff zu ersetzen, der auch in den Vorstandsetagen und Aktionärsversammlungen schmackhaft gemacht werden kann.

Der neue Begriff wurde auf dem Gründungstreffen der Open-Source-Initiative im Februar 1998 geprägt. Daran nahmen John Hall und Larry Augustin (Linux International), Sam Ockman (Silicon Valley Linux User Group), Eric Raymond sowie Todd Anderson und Christine Peterson[133] (Foresight Institute) teil. Peterson erdachte den Begriff "Open Source". Anlaß für dieses Treffen war Netscapes Ankündigung Ende Januar gewesen, den Quellcode seines Browsers offenzulegen. Netscape hatte Raymond eingeladen, ihnen dabei zu helfen. "We realized that the Netscape announcement had created a precious window of time within which we might finally be able to get the corporate world to listen to what we have to teach about the superiority of an open development process. We realized it was time to dump the

---

[129] Führender Distributor dort ist TurboLinux. Zur Verbreitung in Indien s. Harsh 2/1999

[130] Kittler, Wizards 7/1999

[131] Raymond 1998

[132] Für den Begriffswechsel siehe "14. Version and Change History"

[133] http://www.foresight.org/FI/Peterson.html

confrontational attitude that has been associated with `free software' in the past and sell the idea strictly on the same pragmatic, business-case grounds that motivated Netscape."[134]

Bruce Perens, Autor der *Debian Free Software Guidelines* und der davon abgeleiteten *Open Source Definition* (OSD)[135], meldete Open Source als Warenzeichen an. Die Website www.opensource.org wird gestartet. Viele kritisierten den Begriff "Open Source" und zogen das etablierte "Free Software" vor, andere innerhalb der lockeren Bewegung folgten der Wende.[136] So lud der O'Reilly Verlag im April 1998, kurz nachdem Netscape den Quelltextes seines Navigators freigegeben hatte, noch zu einem *Free Software Summit* ein. Im August des Jahres organisierte der Verlag dann den *Open Source Developer Day*. Die Computer-, aber auch die Finanzpresse griff nach Netscapes Schritt den Begriff 'Open Source' auf.

Quelloffenheit ist auch ein definierendes Merkmal von *free software*, doch indem diese neue Fraktion das Merkmal der Freiheit in den Hintergrund rückt, eröffnet sie die Möglichkeit von Software, deren Quellcode einsehbar, aber nicht modifizierbar ist, oder von Softwareunternehmen, die sich vorbehalten, nach firmenstrategischen Gesichtspunkten zu entscheiden, ob sie Modifikationen von Nutzern aufnehmen oder nicht. Die ursprüngliche Vision eines ständig wachsenden, unwiederruflich freien Software-Universums wird durch diesen Schritt verwässert.

Hinter der neuen Sprachpolitik stehen auch Angriffe gegen die Person Stallmans -- eine Art Vatermord. "To the new generation, Stallman is an embarrassment and a hindrance who must, at all costs, be trundled into a back room before he scares off the investors."[137] Andrew Leonard führt in einem Artikel im Salon Magazine Stimmen an, die von einem Generationswechsel sprechen und Kritik an Stallmans Stil äußern, seine Projekte zu leiten.

> "'GNU has floundered over the past few years precisely because of Richard Stallman's hyper-controlling mentality,' wrote Mates. 'The culture of Linux/Apache/etc. development is friendlier to developers: Linus Torvalds loves to work with others, admits his faults and failings, and accepts stuff that's better than his own. Richard Stallman's development philosophy is almost antithetical -- he must control, and so his projects suffer.'"[138]

Der Unterschied im 'Führungsstil' war es auch, der Raymond dazu veranlaßte, GNU als ein Kathedralen-Projekt dem Basar-Modell von Linux und anderen Projekten der dritten Generation gegenüberzustellen. Im selben Artikel zitiert Leonard eine eMail von Raymond, in der er Stallmans Auftritt beim *Open Source Developer Day* kritisiert:

> "When the purpose of the event is to sell our ideas to the trade press and business, there are times when the speeches of people you disagree with are functionally helpful and yours are not. Therefore, if I am trying to get victory for all of us, I may have to put pressure on you but not on the people who disagree with you -- even if my private views are actually closer to yours. ...

---

[134] History of the Open Source Initiative, http://www.opensource.org/history.html

[135] s.u. unter Lizenzen

[136] Perens 1999: 174

[137] Leonard 8/1998

[138] Leonard 9/1998

> You [Stallman] did a lot of damage, more than you probably know. I've seen three different articles with the basic theme ‘those flaky hackers can't get it together and won't grow up,’ every one citing your diatribe. Next time, it's going to be a lot harder for me to argue that you ought to be included. And a lot harder for me to be upset when I lose.”[139]

Stallman selbst sieht durch diesen Begriffswechsel das Gleichgewicht zwischen Verbreitung und Aufklärung bedroht. “Thus, the rhetoric of ‘open source’ focuses on the potential to make high quality, powerful software, but shuns the ideas of freedom, community, and principle.”[140] Die Open-Source-Fraktion betone allein die pragmatischen Aspekte, die Nützlichkeit, die Features, die Zuverlässigkeit und Effizienz der Software. “The approach or the attitude that is associated with Linux is a very different one, a much more technological one, an engineers' attitude: how can we make powerful software, how can we be ‘successful.’ These are things that are not bad things, but they are missing the most important point.”[141] Entsprechend beharrt das GNU-Projekt auf der Bezeichnung ‘free Software’ um zu betonen, daß Freiheit und nicht allein Technologie zählt. “That is the real difference between free software and open source. Free software is a political philosophy and open source is a development methodology -- and that's why I'm proud to be part of the free software movement, and I hope you will be too.”[142]

> “The Free Software movement and the Open Source movement are like two political parties within our community. Radical groups are known for factionalism [...] They agree on the basic principles, and disagree only on practical recommendations. [...] For the Free Software movement and the Open Source movement, it is just the opposite on every point. We disagree on the basic principles, but agree on most practical recommendations. We work together on many specific projects. In the Free Software movement, we don't think of the Open Source movement as an enemy. The enemy is proprietary software. But we do want people in our community to know that we are not the same as them!”[143]

Perens, der mit der *Open Source Definition*[144] die Meßlatte für die Erteilung des Gütesiegels Open Source™ geschaffen und zusammen mit Raymond die Open Source Initiative (OSI) zu seiner Zertifizierung gegründet hatte, wechselte ein Jahr später das politische Lager. Die OSI habe ihre Aufgabe, der Nicht-Hacker-Welt die freie Software nahzubringen, erfüllt. “And now it's time for the second stage: Now that the world is watching, it's time for us to start teaching them about Free Software. Notice, I said Free Software, *not* Open Source.”[145] Er bedauerte, daß Open Source die Bemühungen der FSF überschattet habe -- “a schism between the two groups should never have been allowed to develop.” -- verlies die OSI und wählte wieder die Seite von SPI und FSF.

---

[139] Leonard 9/1998
[140] Stallman 1999: 69 f.
[141] Stallman, Wizards 7/1999
[142] Stallman, Wizards 7/1999
[143] Stallman 1999b
[144] s.u. unter “Lizenzen”
[145] Perens 1999a

Leonards Fazit in der August-Ausgabe von Salon Magazine: “Unkempt and off-kilter though he may be, Stallman embodies the fervor and the faith that make free software worth embracing. If the pragmatists of the open source cause sacrifice him to make free software safe for business, it seems to me, they risk losing their movement's soul.”[146]

Zahlen zur freien Software

Da Freie Software ohne Registrierung installiert und kopiert werden kann, gibt es keine genauen Zahlen über Installationen, Nutzer oder Marktanteile. Für Betriebssysteme und Server gibt es automatische Verfahren, um die benutzte Software von im Internet sichtbaren Host-Rechnern abzufragen, weshalb Näherungszahlen für GNU/Linux, BSD und Apache vorliegen. Für Linux gibt es außerdem eine Selbstregistrierung. Zu den meisten anderen Projekten liegen keinerlei Anhaltspunkte vor. (Zu Markterwartungen für freie Software siehe auch unten unter “Wirtschaftliche Aspekte”)

Eine freiwillige Selbstregistrierung bietet der *Linux Counter*,[147] der im August 2000 auf weit über 15.000 stand. Harald.T.Alvestrand schätzt aufgrund dieser Angabe und weiterer Indizien die weltweite Zahl der GNU/Linux-Nutzer auf 14 Millionen.[148]

Bei den Internet-Servern liegt freie Software auf Platz eins. Der *Internet Operating System Counter*,[149] der leider nicht fortgeführt wird, stellte im April 1999 unter 1.465.124 abgefragten Host mit den Adresses 'ftp.', 'news.' and 'www.' fest, daß weltweit 31,3% (399.748) mit Linux betrieben wurden. Unter der Länderdomäne .de lag der Linux-Anteil bei 42,7% (197.670). Die BSD-Familie lag weltweit bei 14,6% (186.385), in .de bei 8% (36.858).

Netcraft,[150] das einen laufenden Web Server Survey durchführt, gibt für Juni 2000 folgende Zahlen: von 17,3 Millionen erfaßten Hosts lief auf 35.73% (6.116.811) Linux, gefolgt von Microsoft mit 21.32% (3.644.187). Für den Apache erhob Netcraft im Juli 2000 auf einer Basis von 18,2 Millionen Sites einen Anteil von 62.81% (11.412.233), gefolgt von Microsoft-Produkten mit 19.86% (3.608.415). Zu BSD macht Netcraft keine Angaben.

---

[146] Leonard 8/1998

[147] http://counter.li.org/

[148] Harald.T.Alvestrand, Estimating the number of Linux users, http://counter.li.org/estimates.html

[149] http://www.leb.net/hzo/ioscount/

[150] http://www.netcraft.co.uk/survey/

Was ist freie Software, wie entsteht sie, wer macht sie?

Freie Software stellt eine von der proprietären Software grundlegend verschiedene Welt mit ihrer eigenen Kultur dar. Alle weiteren Merkmale ergeben sich aus ihren drei Grundeigenschaften: (1) der Quellcode freier Software ist verfügbar, (2) sie darf ohne Einschränkungen und ohne Zahlungsverpflichtungen kopiert und weitergegeben werden und (3) sie darf verändert und in veränderter Form weitergegeben werden.[151] Diese Eigenschaften bilden die idealen Voraussetzungen für eine offen, d.h., nicht auf Vertragsverhältnissen[152] beruhende, kooperative Software-Entwicklung und eine weitestgehende Beteiligung der Anwender.

Jedes größere Software-Projekt wird von Gruppen von Entwicklern erstellt. Auch in der Industrie haben heute Teams, die über eine kreative Selbständigkeit verfügen, ihren Code synchronisieren und in regelmäßigen Abständen das gemeinsame Produkt stabilisieren, die hierarchischen Verfahren unter einem Chef-Programmierer weitgehend abgelöst. Beobachter wollen selbst bei Microsoft eine hochskalierte Hacker-Methodik im Einsatz sehen.[153] Die freie Software dagegen hat dank der genannten Eigenschaften und dank des Internet die Zusammenarbeit auf alle ausgeweitet, die sich beteiligen möchten. Da hier weder Leistungslohn noch Geheimhaltungsvorschriften die Team-Grenzen bestimmen, kann das entstehende Wissen nur allen gehören.

Quellcode und Objektcode

Software stellt eine besondere Klasse von Wissen dar. Es handelt sich um operative Anweisungen, die, von Menschen geschrieben, sich an einen Computer richten. Menschen schreiben diese Anweisungen in einer höheren Programmiersprache, wie C, C++ oder Pascal. Dieser Ausgangstext oder Quellcode wird einem Compiler übergeben, der ihn in eine maschinennähere, für Menschen nicht mehr lesbare Form, den Objektcode übersetzt. Diese binäre Form des Programms erst veranlaßt den Rechner, für den Menschen (meist) sinnvolle Zeichenoperationen auszuführen.

Kompilierung ist ein im wesentlichen irreversibler Prozeß. D.h., aus dem Binärcode läßt sich allenfalls mit großen Anstrengungen, die der ursprünglichen Codierung nahekommen, der Quellcode rekonstruieren.[154] Proprietäre Software wird nur in einer für eine bestimmte Prozessorplattform vorkompilierten Form ausgeliefert, ohne Quellen. Das stellt zwar keinen Kopierschutz dar, das Wissen darüber, wie ein solches *Black Box*-System macht, was es macht, ist jedoch für alle praktischen Belange effektiv geschützt. Böse Zungen mutmaßen, daß viele Firmen ihre Quellen nicht veröffentlichen, damit niemand sieht, von wie geringer Qualität ihre Software ist, doch der Hauptgrund liegt natürlich im Schutz des

---

[151] diese drei Eigenschaften in einer Vertragsform verbindlich festzuschreiben war die große Innovation der GNU Public License; s.u. unter “Lizenzen”

[152] über die Lizenz hinaus, die sehr wohl ein Vertragsverhältnis zwischen Anbieter und Nutzer darstellt.

[153] Cusumano/Selby 1997

[154] Durch Reverse Engineering, Dekompilierung oder Disassemblierung, die alle in den gängigen Lizenzen proprietärer Software untersagt sind. Eine solche Rückübersetzung ist jedoch nach deutschem Urheberrecht zulässig, sofern sie der Herstellung der Interoperabilität eines unabhängigen Computerprogramms dient (§ 69 e, UrhG).

geistigen Eigentums. Microsoft wehrte sich gegen die Offenlegung des Quellcodes von Windows in einem Rechtsstreit mit Caldera mit der Begründung, es handele sich um “eines der wertvollsten und geheimsten Stücke geistigen Eigentums in der Welt”.

Für reine Anwender mag eine *Black Box* zunächst keine wirkliche Qualitätseinschränkung bedeuten, für Entwickler und Systemadministratoren macht es ihre Arbeitsumgebung zu einer Welt voll Mauern und verbotener Zonen. Für Anpassung, Integration, Fehlerbehebung und Ergänzung müssen sie Veränderungen an der Software vornehmen, und das ist nur am Quelltext möglich. Freie, quelloffene Software bietet ihnen diese Möglichkeit. Bei proprietärer Software bleibt ihnen nur, auf die Unterstützung durch den Hersteller zu hoffen oder sich durch umständliches und nur in engen Grenzen legales *Reverse Engineering* zu behelfen.

Ein häufig von Anbietern proprietärer Software vorgebrachtes Argument lautet, ihre Software umfasse mehrer Millionen Zeilen Code, weshalb ihre Kunden gar kein Interesse hätten, den Quellcode zu lesen. Im übrigen verfalle durch den Eingriff in die Software die Gewährleistungsgarantie, und der Support würde ungleich komplexer.[155]

Das Gegenargument lautet, daß keine Software hundertprozentig das leistet, was Anwender sich wünschen. Anspruchsvolle Anwender werden also in jedem Falle die Möglichkeit schätzen, in den Quellcode eingreifen, oder andere damit beauftragen zu können, die gewünschten Veränderungen vorzunehmen. Selbst wenn keine Änderungen vorgenommen werden sollen, ist es sinnvoll, den Quellcode mitzuliefern, da Interessierte ihn lesen und daraus lernen können.

Modifikationen an proprietärer Software sind technisch nicht möglich und lizenzrechtlich nicht erlaubt. Die Datenformate, die diese Software schreibt, sind häufig nicht dokumentiert, weshalb die Gefahr besteht, daß ein Generationswechsel oder die Einstellung einer Software, z.B. durch Verkauf oder Konkurs des Herstellers, die über Jahre angesammelten Daten unlesbar und unkonvertierbar macht. Auch die urheberrechtlich geschützten Handbücher dieser Software erlauben keine Anpassung und Fortschreibung, um Änderungen oder die an einzelnen Arbeitsplätzen eingesetzten Varianten zu dokumentieren.

In der freien und der proprietären Software stehen sich also zwei grundlegend verschiedene Auffassungen über das Wissen gegenüber. Hier Quelltext als ein in geschlossenen Gruppen, unter Vertraulichkeitsverpflichtung gefertiges Master-Produkt, das in geschlossener, binärer Form als Ware vermarktet und mit Hilfe von Urheberrechten, Patenten, Markenschutz und Kopierschutzmaßnahmen vor Lektüre, Weitergabe und Veränderung geschützt wird. Dort Quelltext als in einer offenen, nicht-gewinnorientierten Zusammenarbeit betriebener Prozeß, bei dem eine ablauffähige Version immer nur eine Momentaufnahme darstellt, deren Studium, Weitergabe und Modifikation die Lizenzen der freien Software ausdrücklich ermutigen. Hier eine Ware, die dem Konsumenten vom Produzenten verkauft wird. Dort ein kollektives Wissen, das allen zur Verfügung steht.

Wie funktioniert ein Projekt der freien Software?

Es beginnt meist damit, daß jemand ein Problem hat. Das Sprichwort ‘Notwendigkeit ist die Mutter aller Erfindungen’ übersetzt Eric Raymond in eine der Faustregeln der freien

[155] So Frank Gessner von Intershop, Wizards 7/1999 Diskussion. Er fügte hinzu, daß Systemintegratoren, Dienstleister oder Design-Agenturen, die darauf bestehen, den Quellcode von Intershop-Software-Systemen erhalten können.

Software: “Every good work of software starts by scratching a deverloper’s personal itch”.[156] So wollte Tim Berners-Lee 1990 die Informationsflut, die am CERN produziert wird, in den Griff bekommen und begann, das WWW-Protokoll zu entwickeln.[157] Rob McCool am NCSA wollte einen Server für dieses Format haben und schrieb NCSA-httpd. David Harris arbeitete an einer Universität in Dunedin, Neuseeland, die 1989 ein Novell NetWare-Netzwerk installierte, nur um festzustellen, daß es kein eMail-System enthielt. Das Budget war aufgebraucht, die kommerziellen eMail-Pakete sehr teuer, also schrieb Harris in seiner Freizeit ein einfaches Programm namens Pegasus Mail, das er seither zu einem mächtigen und komfortablen Werkzeug weiterentwickelt hat und weiterhin verschenkt.[158] Brent Chapman wurde 1992 zum Administrator von 17 technischen Mailinglisten. Mit der damals verfügbaren Software mußte der Listenverwalter jede Subskription und andere Adminstrationsvorgänge von Hand eintragen, was eine Menge Zeit kostete. Deshalb begann er Majordomo zu entwickeln, der heute am weitesten verbreitete Mailinglisten-Server im Internet.[159] Linus Torvalds wollte ein Unix auf seinem 386er PC zuhause laufen lassen, fand allein Minix, das seinen Ansprüchen nicht genügte, und begann Linux zu entwickeln. In einem Interview mit der c't zeigt sich Torvalds noch zehn Jahre später verwundert über den Erfolg.

> “Ich muß sagen, daß ich niemals etwas Vergleichbares erwartet hätte. Nicht einmal annähernd. Ich wollte ein System haben, das ich selbst auf meiner Maschine nutzen konnte, bis es etwas besseres gäbe. Ich habe es im Internet zur Verfügung gestellt, nur für den Fall, daß jemand anderes an so einem Projekt Interesse hätte. Daß es nun Millionen von Anwendern gefunden hat und im Internet ziemlich bekannt ist, hätte ich mir niemals träumen lassen.”[160]

Freie Software entsteht also nicht auf Anweisung eines Vorgesetzten oder Auftraggebers. Sie ist vielmehr eine eigenmotivierte Tätigkeit, angetrieben von dem Wunsch, ein vor der Hand liegendes Problem bei der Arbeit oder Forschung zu lösen. Eine Ausnahme bildet das GNU-Projekt, das von der Vision eines vollständigen freien Betriebssystems geleitet wurde.

All diese Männer der ersten Stunde -- es scheint tatsächlich nicht eine einzige Frau unter ihnen zu geben[161] -- veröffentlichten ihre Projekte frühzeitig, in der Annahme, daß sie nicht die einzigen sind, die dieses Problem haben, und daß es andere gibt, die ihnen bei der Lösung helfen würden.

Meist bildet sich schon bald nach der Erstveröffentlichung um den Initiator eine Gruppe von Mitstreitern, die ihre Kenntnisse und Zeit in das Projekt zu investieren bereit

---

[156] Raymond 1998

[157] vgl. Berners-Lee 1999

[158] “History of Pegasus Mail” aus dem Help-Menü, http://www.pmail.com. Harris bezeichnet Pegasus als “free software”, im technischen Sinne handelt es sich jedoch um Freeware. Es kann kostenlos weiterverbreitet, aber nicht modifiziert werden, und es ist nicht zu einem kooperativen Projekt geworden. S. “Terms and Conditions”, ebd.

[159] Chapman 1992

[160] nach Sommerfeld 1999: Kap. 7 “Linux wird bekannt”

[161] vgl. Sollfrank 1999. Sollfrank beginnt ihre Untersuchung mit einer differenzierten Definition von ‘Hacker’ und ‘Cracker’, forscht dann aber nur nach weiblichen Crackern, um festzustellen (mit einer einzigen Ausnahme): “there are NO women hackers.” Die Evidenz z.B. auf Konferenzen der freien Software bestätigt jedoch die Vemutung, daß es sich um eine vorwiegend männliche Beschäftigung handelt.

sind und die, wenn die Zahl der Beteiligten wächst, als *Maintainer* eine koordinierende Verantwortung für Teile des Projekts übernehmen.

Core-Team und Maintainer

Wenn die Zahl der Mitentwickler und Anwender wächst, bildet diese Gruppe das zentrale Steuerungs-Gremium des Projekts, das *Core-Team*. Ein solches Team rekrutiert sich meist aus den Leuten, die entweder schon am längsten daran arbeiten oder sehr viel daran gearbeitet haben oder derzeit am aktivsten sind. Beim Apache umfaßt es 22 Leute aus sechs Ländern. XFree86 hat ein *Core-Team* von elf Personen und eine *Community* von etwa 600 Entwicklern. Innerhalb des Core-Teams werden Entscheidungen gefällt über die allgemeine Richtung der Entwicklung, über Fragen des Designs und interessante Probleme, an denen weitergearbeitet werden soll.

Große Projekte werden in funktionale Einheiten, in *Packages* oder Module gegliedert, für die jeweils ein oder mehrere *Maintainer* zuständig sind. An jedem Einzelprojekt arbeiten mehrere Dutzend bis hundert Entwickler weltweit mit. Änderungen werden an das *Core-Team* geschickt und von diesem in den *Source-Baum* integriert.

Beim Linux-Kernel gibt es kein offizielles Entwicklerteam. Im Laufe der Zeit hat sich meritokratisch eine Gruppe von fünf oder sechs Leuten herausgeschält, die das Vertrauen des zentralen Entwicklers Linus Torvalds genießen. Dazu gehört Dave Miller, der sich um den SPARC-Port kümmert. Alles, was das Dateisystem betrifft, geht an Stephen Tweedie, und Alan Cox ist zuständig für die allgemeinen Aspekte der nächsten Kernel-Version. Sie sammeln die eintreffenden *Patches*, testen sie und geben sie vorgefiltert an Torvalds weiter. Er trifft dann aufgrund der Vorentscheidungen seiner Vertrauten die letzte Entscheidung.[162]

Die Community

Der *Community* messen die neueren Projekten, wie Linux und Apache, einen größeren Stellenwert bei, als das ältere GNU-Projekt. Im *GNU Manifesto* von 1985 schrieb Stallman: “GNU ... is ... the software system which I am writing so that I can give it away free ... Several other volunteers are helping me.”[163] In einem Interview im Januar 1999 sagte er auf die Frage, wieviele Menschen im Laufe der Jahre an GNU mitgearbeitet haben: “Es gibt keine Möglichkeit, das festzustellen. Ich könnte vielleicht die Anzahl der Einträge in unserer Freiwilligendatei zählen, aber ich weiß nicht, wer am Ende wirklich etwas beigetragen hat. Es sind wahrscheinlich Tausende. Ich weiß weder, wer von ihnen echte Arbeit geleistet hat, noch ist jeder, der echte Arbeit geleistet hat, dort aufgeführt.”[164] Bei den jüngeren Projekten wird stärker auf das (im übrigen auch urheberpersönlichkeitsrechtlich verbriefte) Recht auf Namensnennung geachtet. Der Pflege der *Community*, die die Basis eines Projektes bilden, wird mehr Aufmerksamkeit gewidmet.

Das Debian-Team besteht aus etwa 500 Mitarbeitern weltweit. Bei XFree86 sind es rund 600. Bei den meisten Open Source-Projekten gibt es keine festgelegte

---

[162] Dietrich 1999

[163] Stallman 1985

[164] Interview mit Richard Stallman von Robin Hohmann, in Computerwoche, Nr. 4 vom 29. Januar 1999, http://www.computerwoche.de/info-point/heftvorschau/detail.cfm?SID=5556

Aufgabenverteilung. Jemand macht das, was ihn interessiert, und implementiert und programmiert das, wozu er Lust hat.

Auch der Apache-Webserver wird natürlich nicht allein von den 22 Core-Team-Mitgliedern entwickelt. Es gibt sehr viele Leute, die -- regelmäßig, gelegentlich, zum Teil auch nur einmalig -- Fehler im Apache beheben. Die Beteiligung fängt an bei simplen *Bug Reports* und geht über *Feature Requests*, über Ideen zur Weiterentwicklung bis hin zu *Patches* oder größeren Funktionserweiterungen, die von Nutzern gemacht werden, die ebenfalls Spaß am Entwickeln haben und ihren Teil dazu beitragen wollen, daß der Apache auch weiterhin der beste Webserver bleibt.[165]

Auch das freie X-Window-System beruht auf einem solchen Spektrum vielfältiger Mitarbeit.

> "Die Anzahl der neuen Leute, die Interesse zeigen und in so ein Projekt einsteigen, ist absolut verblüffend. Das Interesse ist riesig. 'Entwickler' ist natürlich ein bißchen grob gegriffen. Die Zahl der Leute, die mehr als 1000 Zeilen Code im XFree86 drin haben, ist vielleicht zwei Dutzend, mehr sind das nicht. Es gibt jede Menge Leute, die mal einen ein- bis dreizeiligen *Bug Fix* gemacht haben, aber auch 'nur' Tester. Leute, die Dokumentationen schreiben, sind wahnsinnig wertvoll. Denn, das werden viele Free Software-Projektleiter mir bestätigen können: *Coders don't write docs*. Es ist leicht, Leute zu finden, die genialen Code schreiben. Es ist schwierig, Leute zu finden, die bereit sind, diesen genialen Code auch für den Anfänger lesbar zu dokumentieren."[166]

Entscheidungsfindung: "rough concensus and running code"

Das Credo der Internet-Entwicklergemeinde lautet: "We reject: kings, presidents and voting. We believe in: rough concensus and running code."[167] Die gleiche Philosophie herrscht auch in den meisten freien Software-Projekten. Auch die *Core-Team*-Mitglieder sind keine Projektleiter oder Chefs. Lars Eilebrecht über Apache:

> "Alle Entscheidungen, die getroffen werden, sei es nun welche Patches, welche neuen Funktionalitäten in den Apache eingebaut werden, was in der Zukunft passieren soll und sonstige Entscheidungen werden alle auf Basis eines Mehrheitsbeschlusses getroffen. Das heißt, es wird auf der Mailingliste darüber abgestimmt, ob ein bestimmter Patch eingebunden wird oder nicht. Bei Patches, die eine große Änderung darstellen, ist es typischerweise so, daß sich mindestens drei Mitglieder der Apache-Group [des *Core-Teams*] damit beschäftigt haben müssen, das heißt, es getestet haben und dafür sein müssen, daß der Patch eingebaut wird. Und es darf keinerlei Gegenstimmen geben. Wenn es sie gibt, dann wird typischerweise das Problem, das jemand damit hat, behoben und, wenn der Patch dann für sinnvoll erachtet wird, irgendwann eingebaut."[168]

---

[165] Lars Eilebrecht, Apache-Core-Team-Mitglied, Wizards 7/1999

[166] Dirk Hohndel, Wizards 7/1999

[167] Dave Clark, IETF Credo (1992), http://info.isoc.org:80/standards/index.html

[168] Eilebrecht, Wizards 7/1999

Selbstverständlich gibt es auch in diesen Projekten Meinungsverschiedenheiten, doch anders als bei philosophisch-politischen Debatten, die oft auf Abgrenzung und Exklusion von Positionen zielen, scheinen Debatten über technische Fragen eher zu inklusiven Lösungen zu neigen. Wirkliche Zerreißproben hat Dirk Hohndel in den Projekten, in denen er in den letzten acht Jahren beschäftigt war, noch nicht erlebt.[169]

Das offen-kooperative Verhältnis ändert sich auch nicht notwendig dadurch, daß Firmenvertreter an Entscheidungsprozessen teilnehmen. Auch im *Core-Team* des Apache-Projekts gibt es, wie Eilebrecht berichtet, gelegentlich Konflikte.

> "Dabei ist es aber vom Prinzip her uninteressant, ob die Person, die damit irgendein Problem hat, von einer Firma ist oder ob sie privat bei der Apache-Group dabei ist. Wir haben zwar vom Prinzip her, ich will nicht sagen Kooperationen, aber Firmen mit in der Apache-Group dabei, aber in Form eines Vertreters dieser Firma. Das heißt z.B., IBM ist mit dabei, Siemens und Apple demnächst. Aber sie haben im Prinzip bei Abstimmungen, oder wenn irgendetwas entschieden werden soll, jeweils nur eine Stimme. Im Fall von IBM sind es mittlerweile zwei, aber das ist ein anderes Thema. Aber, wenn die meisten anderen Mitglieder etwas dagegen haben, daß bestimmte Änderungen gemacht werden oder bestimmte Entscheidungen nicht gewollt sind, dann haben die keine Chance, das irgendwie durchzusetzen. Dann müssen sie entweder aussteigen oder akzeptieren, daß es nicht gemacht wird."[170]

Code-Forking

Das eben Gesagte bedeutet nicht, daß Konflikte prinzipiell nicht dazu führen können, daß Fraktionen getrennte Wege gehen. Tatsächlich hat es verschiedentlich Spaltungen von Projekten gegeben. Im schlimmsten Fall bedeutet dies den Tod eines Projekts, oder es entstehen daraus zwei ähnliche Projekte, die um Entwickler und Anwender konkurrieren, und die Gesamtbemühungen verdoppeln sich. Im günstigsten Fall kann eine Spaltung fruchtbar sein. Die entstehenden Einzelprojekte entwickeln die Software für verschiedene komplementäre Anwendungsschwerpunkte weiter, ergänzen sich und profitieren von Innovationen in den anderen Projekten (z.B. FreeBSD, NetBSD und OpenBSD; s.u.).

Die Möglichkeit, sich jederzeit von einem Projekt abzusetzen und es in eine eigene Richtung weiterzutreiben, wird auch als heilsam erachtet. Es verhindert, daß unwartbare Mammutprogramme entstehen und Personen, die Verantwortung für Programme tragen, sich nicht mehr darum kümmern. Beispiel ist der GNU-C-Compiler (GCC), der nach einer Spaltung zum EGCS wurde. Die beiden Projekte sind inzwischen wieder unter dem Namen GCC zusammengeführt worden. Das Beispiel Emacs und XEmacs ist ebenfalls zu nennen. Auch im GIMP-Projekt startete Ende 1998 einer der Entwickler eine eigene Verzweigung und lud andere Entwickler ein, nun an seiner Variante des Grafikprogramms mitzuarbeiten. Die Spaltung konnte jedoch abgewehrt und der Code der Zweigentwicklung in das Hauptprojekt integriert werden.

Die Tools

---

[169] Hohndel, Wizards 7/1999 Diskussion

[170] Eilebrecht, Wizards 7/1999 Diskussion

Die zentralen Kommunikationsmittel für die weltweit ortsverteilte Kooperation sind Email, genauer Mailinglisten, sowie Newsgroups. Für Echtzeitkommunikation verwenden einige Projekte auch den *Internet Relay Chat* (IRC). Die Projekte präsentieren sich und ihre Ressourcen auf Websites. Das zentrale Instrument zur kooperativen Verwaltung des Quellcodes sind CVS-Server.

Das *Concurrent Versions System* (CVS) ist ein mächtiges Werkzeug für die Revisionsverwaltung, das es Gruppen von Entwicklern erlaubt, die gleichzeitige Arbeit an denselben Dateien zu koordinieren und einen Überblick über die Veränderungen zu behalten. CVS ist Standard bei freier Software, aber auch viele Firmen, die kommerziell Software erstellen, setzen es ein. Jeder kann lesend weltweit auf die Source-Bäume zugreifen und sich die aktuellste Version einer Software auf seine lokale Festplatte kopieren.

Wer die Software fortschreiben möchte, muß sich registrieren, um für andere Entwickler ansprechbar zu sein. Die Sammlung von Dateien liegt in einem gemeinsamen Verzeichnis, dem *Repository*. Um mit der Arbeit zu beginnen, führt man den *Checkout*-Befehl aus, dem man den Verzeichnispfad oder den Namen des Moduls übergibt, an dem man arbeiten möchte. Das CVS kopiert dann die letzte Fassung der gewünschten Dateien aus dem Repository in einen Verzeichnisbaum auf der lokalen Festplatte. Der Entwickler kann diese Dateien nun mit einem Editor seiner Wahl verändern, sie in eine Output-Datei ‘bauen’ (*build*) und das Ergebnis testen. Ist er mit dem Ergebnis zufrieden, schreibt er es mit dem *Commit*-Befehl in das Repository zurück und macht damit seine Änderungen anderen Entwicklern zugänglich. Die neue Version kann jetzt mit älteren verglichen, Patch-*Diffs* können aus einer Basisrevision erstellt und mit den Änderungen anderer Entwickler verschmolzen, nicht mehr benötigte Dateien gelöscht und Information über Dateien abgefragt werden.

Wenn andere Entwickler zur selben Zeit dieselben Dateien bearbeiten, werden die verschiedenen neuen Versionen lokal mit dem *update*-Befehl verschmolzen. Läßt sich das Ergebnis korrekt bauen und testen, werden die verschmolzenen Dateien gleichfalls in den CVS-Baum zurückgeschrieben. Ist das nicht automatisch möglich, müssen sich die beteiligten Entwickler über die Integration ihrer Änderungen untereinander verständigen. Diese als *copy-modify-merge* bezeichnete Methode hat den Vorteil, kein *Lock* der Quelldateien zu erfordern, d.h., Dateien, die gerade von einem Entwickler bearbeitet werden, sind nicht für alle anderen gesperrt. In regelmäßigen Abständen, gewöhnlich wenn bestimmte Meilensteine in der Entwicklung erreicht sind, werden Zweige des Quellbaums mit einem besonderen *Tag* versehen und damit für die folgende *Release* gekennzeichnet.[171]

Debugging

Software enthält Fehler. Einer Informatiker-Faustregel zufolge einen pro hundert Zeilen Code. Sich dieser Tatsache des Lebens öffentlich zu stellen, fällt Software-Unternehmen schwer. Bei ihnen herrscht, was Neal Stephenson eine “institutionelle Unehrlichkeit” nennt.

> “Commercial OSes have to adopt the same official stance towards errors as Communist countries had towards poverty. For doctrinal reasons it was not possible to admit that poverty was a serious problem in Communist countries, because the whole point of Communism was to eradicate poverty. Likewise, commercial OS

[171] für Information über CVS im CVS-Format s. http://www.loria.fr/~molli/cvs-index.html

> [Operating System] companies like Apple and Microsoft can't go around admitting that their software has bugs and that it crashes all the time, any more than Disney can issue press releases stating that Mickey Mouse is an actor in a suit. ... Commercial OS vendors, as a direct consequence of being commercial, are forced to adopt the grossly disingenuous position that bugs are rare aberrations, usually someone else's fault, and therefore not really worth talking about in any detail."[172]

Freie Software-Projekte dagegen können sich dem Problem offen stellen. Ihr Verfahren, mit Bugs umzugehen, stellt einen der wichtigsten Vorteile gegenüber proprietärer Software dar. Die Stabilität dieser Software, d.h. ihr Grad an Fehlerfreiheit, verdankt sich nicht der Genialität ihrer Entwickler, sondern der Tatsache, daß jeder Anwender Fehler an den Pranger stellen kann und die kollektive Intelligenz von hunderten von Entwicklern meist sehr schnell eine Lösung dafür findet.

Wer z.B. in Debian GNU/Linux einem Bug begegnet, schickt einen eMail-Bericht darüber an submit@bugs.debian.org. Der Bug erhält automatisch eine Nummer (Bug#nnn), sein Eingang wird dem Anwender bestätigt und er wird auf der Mailingliste debian-bugs-dist gepostet. Hat der Einsender den Namen des Pakets, in dem der Bug aufgetreten ist, angegeben, erhält auch der Maintainer dieses Pakets eine Kopie. Betrifft der Fehler eines der in der Debian-Distribution enhaltenen selbständigen Pakete (XFree86, Apache etc.), erhalten auch die Zuständigen für dieses Projekt eine Kopie. In das "Reply-to"-Feld der eMail wird die Adresse des Einsenders und nnn@bugs.debian.org eingetragen, so daß die Reaktionen darauf an alle Interessierten in der Projekt-Community gehen. Übernimmt der Maintainer oder ein anderer Entwickler die Verantwortung für die Lösung des Problems, wird auch seine Adresse hinzugefügt. Er ordnet den Bug einer von sechs Dringlichkeitskategorien (*critical, grave, important, normal, fixed* und *wishlist*) zu. Gleichzeitig wird der Bug in die Web-basierte, öffentlich einsehbare Datenbank http://www.debian.org/Bugs eingetragen. Die Liste ist nach Dringlichkeit und Alter der offenen Bugs sortiert und wird einmal pro Woche auf debian-bugs-reports gepostet.[173]

In vielen Fällen erhält man binnen 24 Stunden einen *Patch*, der den Fehler beseitigt, oder doch zumindest Ratschläge, wie man ihn umgehen kann. Auch diese Antworten gehen automatisch in die Bug-Datenbank ein, so daß andere Nutzer, die mit demselben Problem konfrontiert werden, hier die Lösungen finden.

Die Stärke der freien Projekte beruht also darin, daß alle Änderungen an der Software eingesehen werden können, sobald sie in den CVS-Baum eingetragen sind, und daß Releases in kurzen Abständen erfolgen. Dadurch können sich tausende von Nutzern am Auffinden von Bugs und hunderte von Entwicklern an ihrer Lösung beteiligen. Debugging kann hochgradig parallelisiert werden, ohne daß der positive Effekt durch erhöhten Koordinationsaufwand und steigende Komplexität konterkariert würde. In der Formulierung von Raymond lautet diese Faustregel: "Given enough eyeballs, all bugs are shallow."[174] Gemeint ist nicht etwa, daß nur triviale Bugs auf diese Weise beseitigt werden könnten, sondern daß durch die große Zahl von Beteiligten die Chance steigt, daß einige von ihnen genau in dem fraglichen Bereich arbeiten und relativ mühelos Lösungen finden können.

---

[172] Stephenson 1999

[173] Die meisten freien Projekte verwenden ein solches Bug-Tracking-System. S. z.B. die Datenbank der KDE-Bug-Reports: http://bugs.kde.org/db/ix/full.html.

[174] Raymond 1998

Frank Rieger weist darauf hin, daß dieses Verfahren nicht nur durch seine Geschwindigkeit der Problembehebung in der proprietären Software überlegen ist, sondern auch durch die Gründlichkeit seiner Lösungen:

> "[D]adurch, daß die Sourcen offen sind, werden oft genug nicht nur die Symptome behoben, wie wir das z.B. bei Windows-NT sehen, wo ein Problem im Internet-Information-Server ist und sie sagen, wir haben dafür jetzt gerade keinen Fix, weil das Problem tief im Kernel verborgen liegt und da können wir jetzt gerade nichts tun, sondern wir geben euch mal einen Patch, der verhindert, daß die entsprechenden Informationen bis zum Kernel durchdringen -- also: Pflaster draufkleben auf die großen Löcher im Wasserrohr. Das passiert halt in der Regel bei Open Source-Systemen nicht, da werden die Probleme tatsächlich behoben."[175]

Die Geschlossenheit des proprietären Modells ist nützlich für den Schutz des 'geistigen Eigentums', für die Stabilität der Software ist es ein struktureller Nachteil, durch den es nicht mit freier Software konkurrieren kann.

> "In the world of open source software, bug reports are useful information. Making them public is a service to other users, and improves the OS. Making them public systematically is so important that highly intelligent people voluntarily put time and money into running bug databases. In the commercial OS world, however, reporting a bug is a privilege that you have to pay lots of money for. But if you pay for it, it follows that the bug report must be kept confidential -- otherwise anyone could get the benefit of your ninety-five bucks!"[176]

Natürlich gibt es auch in der kommerziellen Software-Welt Testverfahren und Bug-Reports. Nach einer ersten Testphase, die *in-house* mit einer begrenzten Gruppe von Mitarbeitern durchgeführt wird, geben Unternehmen üblicherweise Beta-Versionen heraus. Diese enthalten natürlich keinen Quellcode. Nur das Finden, nicht aber das Beheben von Fehlern wird von den Beta-Testern erwartet. Für diese unentgeltliche Zuarbeit müssen die Tester auch noch selbst bezahlen. Die Beta-Version von Windows2000 beispielsweise verkauft Microsoft in Deutschland für etwa 200 DM. Wer Bug-Reports einsendet, erhält als Dank z.B. ein MS-Office-Paket geschenkt und bekommt einen Nachlaß beim Kauf der 'fertigen' Version. Käufer von Betatest-Versionen sind neben leidenschaftlichen Anwendern, die sofort alles haben möchten, was neu ist, vor allem Applikationsentwickler, die darauf angewiesen sind, daß ihre Programme mit der neuen Version von Windows zusammenarbeiten. Kommt schließlich die erste 'fertige' Version der Software auf den Markt, stellt der Kundendienst eine weitere Quelle für den Rücklauf von Fehlern dar.

Die 95$, von denen Stephenson spricht, sind der Preis, für den Kunden bei Microsoft nach dem "Pay Per Incident"-System einen 'Zwischenfall' melden können. Seit er seinen Aufsatz schrieb, ist der Preis auf 195$ für Zwischenfälle, die über das Internet gemeldet werden, und 245$ für solche, die per Telefon durchgegeben werden, gestiegen. Dafür erhält der Kunde innerhalb von 24 Stunden eine Antwort von einem Microsoft Support

---

[175] Frank Rieger, Wizards 7/1999

[176] Stephenson 1999

Professional.[177] Die Fehlerhaftigkeit der Software sehen solche Unternehmen als zusätzliche Einnahmequelle an.

Die Releases

Durch die beschleunigten Innovationszyklen in der von der wissenschaftlich-technologischen Wissensproduktion vorangetriebenen ‘nachindustriellen Gesellschaft’ (Daniel Bell) werden auch materielle Produkte in immer kürzeren Abständen durch neue Versionen obsolet gemacht. Nirgends ist diese Beschleunigung deutlicher als in der Informations- und Kommunikationstechnologie und hier besonders in der Software. Nach der herkömmlichen Logik kommt eine Software auf den Markt, wenn sie ‘ausgereift’ ist, tatsächlich jedoch, wenn der von der Marketing-Abteilung bekanntgegebene Termin gekommen ist. Freie Projekte dagegen veröffentlichen Entwicklerversionen frühzeitig und in kurzen Abständen. Offizielle Versionen werden dann released, wenn sie den gewünschten Grad von Stabilität erreicht haben. Das Ausmaß der Änderung läßt sich an den Versionsnummern ablesen. Eine Veränderung von Ver 2 auf Ver 3 markiert ein fundamental neues Design, während ein Wechsel von Ver 3.3.3.1 zu Ver 3.3.4 einen kleineren Integrationsschritt darstellt.

Freie Software wird heute, wie mehrfach angesprochen, nicht als Produkt, sondern als kontinuierlicher Prozeß verstanden. Zahlreiche Projekte haben bewiesen, daß eine Entwicklergemeinschaft ihre Software über lange Zeit zuverlässig und in hoher Qualität unterstützen und weiterentwickeln kann. In der amerikanischen Debatte ist von einer ‘evolutionären’ Software-Genese von Mutation, Selektion und Vererbung die Rede. Der Basar sorge dafür, daß in einem ‘Darwinistischen Selektionsprozeß’ die bestangepaßte und effizientest Software überlebe.[178]

Abhängig von der Geschwindigkeit, mit der ein Projekt sich verändert, werden täglich oder im Abstand von Wochen oder Monaten neue Releases herausgegeben.[179] Im Entwickler-Source-Baum werden laufend neue Patches eingegeben, die über das CVS abrufbar sind. Die jeweils avancierteste Fassung einer Software ist naturgemäß noch nicht sehr stabil und daher nicht für den täglichen Einsatz geeignet. Sind die für eine neue Version gesteckten Ziele erreicht, wird der Source-Baum für weitere Änderungen gesperrt. Erst nachdem alle Änderungen weithin getestet und korrigiert worden sind und zuverläßig mit den vorbestehenden Bestandteilen zusammenarbeiten, wird ein offizielles Release freigegeben.

Im Unterschied zu den auftrags- und termingebundenen Entwicklungsarbeiten für proprietäre Software können bei freier Software die Ideen bis ins Detail ausdiskutiert und optimiert werden. Dirk Hohndel (XFree86) sieht darin eine der Stärken von freien Software-Projekten. “Denn ich kann einfach sagen: 'Naja, das funktioniert noch nicht. Ich release das Ding nicht.' Wenn ich jetzt natürlich schon 25 Millionen US-Dollar ausgegeben habe, um den 4. Mai als den großen Release-Tag in der Presse bekannt zu machen, dann sagt

---

[177] zu Microsofts Pay Per Incident-System, das übrigens nur in den USA und Kanada angeboten wird, s. http://support.microsoft.com/support/webresponse/ppi/ppifaq.asp. Eine Liste nicht etwa der offenen, sondern nur der behobenen Bugs in Windows NT 4.0 findet sich unter http://support.microsoft.com/support/kb/ARTICLES/Q150/7/34.asp.

[178] z.B. Seiferth 1999; s.a. Hetze 1999: 3

[179] Bei älteren Projekten machen tägliche Releases keinen Sinn, doch z.B. in der Frühphase des Linux-Kernels veröffenliche Torvalds täglich neue Versionen.

mein Marketing-Department mir: 'Das ist mir wurscht. Du released das Ding.'"[180] Mit der zunehmenden Integration freier Software in konventionelle Wirtschaftsprozesse könnte der Druck jedoch wachsen, angekündigte Release-Termine einzuhalten, um an anderen Stellen die Planbarkeit zu erhöhen. Als z.B. Linus Torvalds in seiner Keynote-Ansprache auf der LinuxWorld im Februar 2000 in New York ankündigte, daß sich die Veröffentlichung des Linux-Kernels 2.4 um ein halbes Jahr verschieben werde, wertete die IT-Fachpresse dies als einen Mangel.[181]

Institutionalisierung: Stiftungen und nichtprofitorientierte Unternehmen

Freie Projekte entstehen als formloser Zusammenschluß von Individuen. Da unter den Beteiligten keinerlei finanzielle oder rechtliche Beziehungen bestehen, stellt dies für das Innenverhältnis kein Problem dar. Sobald jedoch die Außenverhältnisse zunehmen, sehen die meisten Projekte es als vorteilhaft, sich eine Rechtsform zu geben. Anlaß dazu können Kooperationen mit Firmen sein, die z.B. unter einer Vertraulichkeitsvereinbarung Hardware-Dokumentation bereitstellen, so daß Treiber für Linux entwickelt werden können, die Beteiligung an Industriekonsortien, die Möglichkeit, Spenden für Infrastruktur (Server) und Öffentlichkeitsarbeit entgegenzunehmen, oder die Notwendigkeit, Warenzeichen anzumelden oder die Interessen des Projekts vor Gericht auszufechten. Daher bilden sich meist parallel zu den weiterhin offenen Arbeitsstrukturen formelle Institutionen als Schnittstellen zur Wirtschaft und zum Rechtssystem.

Am augenfälligsten wird die Problematik an einer Episode aus dem April 1998. Damals trat IBM an die Apache-Gruppe mit dem Wunsch heran, ihren Server als Kernstück in IBMs neues eCommerce-Paket "WebSphere" aufzunehmen. Es handelte sich um ein Geschäft mit einem Marktwert von mehreren Millionen Dollar, und IBM war gewillt, die Apache-Entwickler dafür zu bezahlen. Doch waren sie erstaunt festzustellen, daß diese nur aus zwanzig über die ganze Welt verstreuten Individuen ohne eine Rechtsform bestand. "So let me get this straight," zitierte Forbes Magazine einen der IBM-Anwälte "We're doing a deal with ... a Web site?"[182] Für eine Geldzahlung von IBM hätte es also gar keinen Empfänger gegeben. Die Apache-Gruppe willigte in das Geschäft ein, unter der Bedingung, daß der Webserver weiterhin im Quellcode frei verfügbar bleiben müsse. IBM zeigte sich in der Währung der freien Software-Szene erkenntlich: mit einem Hack. IBM-Ingenieure hatten einen Weg gefunden, wie der Webserver auf Windows-NT schneller laufen kann. Diese Software gaben sie im Quellcode zur freien Verwendung an die Szene weiter. Aufgrund dieser Erfahrung gründete sich aus dem Apache-*Core-Team* die *Apache Software Foundation* (ASF).[183] Die ASF ist ein mitgliederbasiertes gemeinnütziges Unternehmen. Individuen, die ihr Engagement für die kollaborative Open-Source-Software-Entwicklung unter Beweis gestellt haben, können Mitglied werden. Hauptzweck der ASF ist die administrative Unterstützung der freien Entwicklungsprojekte.[184]

---

[180] Hohndel, Wizards 7/1999

[181] z.B. Inforworld, 3.2.2000, http://www.infoworld.com/articles/pi/xml/00/02/03/000203piibmlinux.xml

[182] Forbes Magazine, 8/1998

[183] http://www.apache.org/foundation/

[184] Bylaws der ASF: http://www.apache.org/foundation/bylaws.html

Die erste rechtskräftige Institution entstand 1985 aus dem GNU-Projekt. Die *Free Software Foundation* (FSF)[185] kann als gemeinnützige Einrichtung steuerlich abzugsfähige Spenden entgegennehmen. Ihre Haupteinnahmen stammen jedoch aus dem Verkauf von freier GNU-Software (damals auf Datenbändern, heute auf CD), von freien Handbüchern und von Paraphernalia wie T-Shirts. Die Gewinne werden benutzt, um einzelne Entwicklungsprojekte zu fördern. So wurden FSF-Mitarbieter dafür bezahlt, die GNU-C-Library und die Bash-Shell zu entwickeln. Rechtsschutz (z.B. bei Lizenzstreitigkeiten) und Öffentlichkeitsarbeit sind weitere Aufgaben der FSF.[186]

Software in the Public Interest (SPI)[187] dient als Dachorganisation für verschiedene Projekte, darunter die Debian GNU/Linux-Distribution, Berlin (ein Windowing-System), Gnome, die Linux Standard Base, Open Source.org und Open Hardware.org.[188] 1997 gegründet, erhielt SPI Inc. im Juni 1999 den Status der Gemeinnützigkeit. Spenden werden verwendet, um Domain-Namen (z.B. debian.org) zu registrieren, CDs für das Testen neuer Releases zu brennen oder Reisekosten für Treffen und Konferenzen zu unterstützen. Auch Hardware-Spenden und Netz-Ressourcen sind immer willkommen. SPI meldet ferner Trademarks (TM) and Certification Marks (CM) an. SPI ist Eigentümer der CMs “Open Source” und “Open Hardware” und des TM “Debian”, wobei die Verwaltung der Rechte den einzelnen Projekten obliegt.

Beim XFree86-Projekt entstand die Notwendigkeit einer Rechtsform daraus, daß das X-Consortium, in dem die industrieweite Standardisierung und Weiterentwicklung des X-Window-Systems verhandelt wird, nur Unternehmen als Mitglied aufnimmt. Das 1992 gegründete Projekt hatte Ende 1993 die Wahl, unter dem Dach einer Organisation teilzunehmen, die bereits Mitglied des Konsortiums war, oder selbst eine Rechtsform zu etablieren. Nachdem sich ein Anwalt fand, der bereit war, eine Unternehmensgründung pro bono vorzunehmen, wählte man den zweiten Weg. Durch das aufwendige Anerkennungsverfahren dauerte es bis Mai 1994, bis das gemeinnützige Unternehmen, die XFree86 Project, Inc.,[189] gegründet war. Als wissenschaftliche Organisation dient sie der Forschung, Entwicklung und Implementation des X-Window-Systems und der Verbreitung von Quellcode, Objektcode, Ideen, Informationen und Technologie für die öffentliche Verwendung.[190]

Die Konstruktion dieser Organisationen ist sorgfältig darauf bedacht, eine Verselbständigung zu verhindern. Ämter werden, ähnlich wie bei NGOs, in der Regel ehrenamtlich wahrgenommen. Die Lizenzen der freien Software sind darauf angelegt, eine proprietäre Schließung der Software zu verhindern. Es ist jedoch noch zu früh, einschätzen zu können, ob Spannungen zwischen der freien Entwickler- und Nutzerbasis und ihren Institutionen und die Herausbildung von bürokratischen Wasserköpfen vermieden werden

---

[185] http://www.fsf.org/fsf/fsf.html

[186] vgl. Richard Stallman, The GNU Project, http://www.gnu.org/gnu/the-gnu-project.html

[187] http://www.spi-inc.org/; By-Laws of SPI Inc., Revision 1, December 10, 1997, http://www.chiark.greenend.org.uk/~ian/spi-bylaws.html

[188] “These projects use the monetary and legal infrastructure available to them through SPI to accomplish their individual goals. SPI serves as a guiding organization only, and does not actively control any of the projects it is affiliated with.” Pressemitteilung 29 Oct 1998, http://www.debian.org/News/1998/19981029

[189] http://www.xfree86.org/legal.html

[190] Bylaws der XFree86 Project, Inc., http://www.xfree86.org/by_laws.html; s.a. Corporate Profile: http://www.xfree86.org/corp_profile.html

können. Gerade durch den Erfolg und die derzeitige Übernahme von Aspekten des freien Software-Modells durch herkömmliche Unternehmen ist die Gefahr nicht von der Hand zu weisen, daß die freie Software ein ähnliches Schicksal erleidet, wie zahlreiche soziale Bewegungen vor ihr.

Die Motivation: Wer sind die Leute und warum machen die das... wenn nicht für Geld?

> "Every decision a person makes stems from the person's values and goals. People can have many different goals and values; fame, profit, love, survival, fun, and freedom, are just some of the goals that a good person might have. When the goal is to help others as well as oneself, we call that idealism.
>
> My work on free software is motivated by an idealistic goal: spreading freedom and cooperation. I want to encourage free software to spread, replacing proprietary software which forbids cooperation, and thus make our society better." (Richard Stallman, Copyleft: Pragmatic Idealism[191])

Unsere Gesellschaften kennen wohltätige, karitative und kulturelle (z.B. Kunstvereine) Formen des Engagements für eine gute Sache, die nicht auf pekuniären Gewinn zielen. Gemeinnützige Tätigkeiten sind steuerlich begünstigt und sozialwissenschaftlich in Begriffen wie *Civil Society*, Dritter Sektor und NGOs reflektiert. Doch während einer Ärztin, die eine Zeit lang in Myanmar oder Äthiopien arbeitet, allgemeine Anerkennung gezollt wird, ist die erste Reation auf unbezahlte Software-Entwickler meist, daß es sich entweder um Studenten handeln muß, die noch üben, oder um Verrückte, da ihre Arbeit in der Wirtschaft äußerst gefragt ist und gut bezahlt würde.

"Jedes Geschäft -- welcher Art es auch sei -- wird besser betrieben, wenn man es um seiner selbst willen als den Folgen zuliebe treibt", weil nämlich "zuletzt für sich Reiz gewinnt", was man zunächst aus Nützlichkeitserwägungen begonnen haben mag, und weil "dem Menschen Tätigkeit lieber ist, als Besitz, ... insofern sie Selbsttätigkeit ist".[192] Diese Humboldtsche Erkenntnis über die Motivlage der Menschen bietet einen Schlüssel für das Verständnis der freien Software.[193] Ihre Entwicklung gründet in einem kreativen Akt, der einen Wert an sich darstellt. Ein *Learning-by-Doing* mag noch von Nützlichkeitserwägungen angestoßen werden, das Ergebnis des Lernens ist jedoch nicht nur ein Zuwachs an Verständnis des Einzelnen, sondern eine Schöpfung, die man mit anderen teilen kann. Statt also aus dem Besitz und seiner kommerziellen Verwertung einen Vorteil zu ziehen, übergeben die Programmierinnen und Programmier der freien Software ihr Werk der Öffentlichkeit, auf daß es bewundert, kritisiert, benutzt und weiterentwickelt wird. Die Anerkennung, die ihnen für ihre Arbeit gezollt wird, und die Befriedigung, etwas in den großen Pool des Wissens zurückzugeben, spielen sicher eine Rolle. Die wichtigste Triebkraft vor allen hinzukommenden Entlohnungen ist die kollektive Selbsttätigkeit.

---

[191] 1998; http://www.gnu.org/philosophy/pragmatic.html

[192] Wilhelm von Humboldt, Ideen zu einem Versuch, die Grenzen der Wirksamkeit des Staates zu bestimmen (1792), zitiert nach: Spinner 1994: 48

[193] Daß externe Belohnung die Motivation nicht nur nicht steigern, sondern senken kann, zeigen verschiedene Untersuchungen, vgl. z.B. Kohn 1987

Entgegen einer verbreiteten Auffassung, daß vor allem Studenten mit viel Freizeit sich für freie Software engagieren, ist das Spektrum erheblich breiter. Bei XFree86 bsw. liegt das Altersspektrum der Beteiligten zwischen 12 und 50 Jahren, und sie leben auf allen Kontinenten der Erde.[194] Möglichkeitsbedingung für die Kooperation, in der freie Software entsteht, ist das Internet. Mit der zunehmenden Verbreitung des Internet wächst somit auch die Zahl derjenigen, die potentiell an solchen Projekten mitarbeiten. Viele sind tatsächlich Studenten (bei XFree etwa ein Drittel, beim GIMP, der von zwei Studenten gestartet wurde, ist es die Mehrzahl). Viele haben eine Anstellung als Programmierer, Systemarchitekt oder Systemsdministrator aber auch in nicht-Computer-bezogenen Berufen und entwickeln in ihrer Freizeit an freien Projekten.

> "Das sind ganz normale Leute, die nach ihrer ganz normalen 60 Stunden-Woche gerne auch noch mal 20, 30 Stunden etwas machen, was Spaß macht. Ich glaube, das entscheidende Kriterium für Leute, die sehr produktiv sind und viel in solchen Projekten arbeiten, ist einfach die Faszination am Projekt. Und diese Faszination bedeutet auch sehr oft, daß man sich abends um sieben hinsetzt, um noch schnell ein kleines Problem zu lösen und auf die Uhr schaut, wenn es vier Uhr früh ist."[195]

Auch die produktivsten Entwickler bei KDE haben einen ganz normalen Tagesjob, oft auch als Programmierer. Für sie ist es eine Frage der Prioritäten. Der Bundebürger verbringe im statistischen Mitteln, so Kalle Dalheimer, 28 Stunden in der Woche vor dem Fernseher. Er selbst besitze keinen und vergnüge sich in dieser Zeit mit Programmieren.[196]

Vom Programmieren fasziniert zu sein, ist natürlich eine Grundvoraussetzung für die Beteiligung an freien Projekten. Der Computer als die universelle Maschine bietet ein großes Spektrum möglicher Kreativität. In ihm etwas Nützliches zu schaffen und von ihm ein instantanes Feedback zu bekommen, kann etwas sehr Erfüllendes sein. Hinzu kommt, daß die Arbeit in einer offenen, konstruktiven Szene stattfindet, die guten Beiträgen bereitwillig Anerkennung zollt. Die Aufmerksamkeit, das Image, der Ruhm und die Ehre, die in dieser meritokratischen Wissensordnung zu erlangen sind, werden häufig als Antriebskraft genannt. Lob aus dieser Szene trägt fraglos dazu bei, das Selbstwertgefühl zu erhöhen. Sebastian Hetze weist jedoch darauf hin, daß noch vor allen Eitelkeiten schon Neugier und Lernbegierde eine hinreichende Motivation darstellen:

> "Wenn ich was dazu tue, dann veröffentliche ich nicht nur für mein Ego, sondern ich setze mich auch der Kritik aus. Und da ist es ganz klar so, daß in der Welt der freien Software ein sehr positives, ein konstruktives Kritikklima herrscht, d.h., ich werde ja für einen Fehler -- Software hat Bugs --, den ich in einem Programm gemacht habe, nicht runtergemacht. Es sind ganz viele Leute da, die versuchen, Fehler mit mir gemeinsam zu beheben, d.h., ich lerne. Und das ist eine ganz wichtige Motivation für sehr viele Menschen, die sich einfach auch daran weiterentwickeln wollen. An dieser Stelle sind die Motivationen ganz offenbar so stark, daß sie über das reine, einfach Monetäre hinausgehen. Ich muß vielleicht sogar an dieser Stelle diese Ego-Präsentation ein bißchen zurücknehmen. Es ist bei den großen Software-Projekten so,

---

[194] Hohndel, Wizards 7/1999-Diskussion

[195] Hohndel, Wizards 7/1999 Diskussion

[196] Dalheimer, Wizards 7/1999-Diskussion

daß natürlich oben immer welche im Rampenlicht stehen, aber die vielen hundert Leuten, die nichts weiter machen, als mal einen Bugfix oder eine gute Idee reinzugeben, kriegen niemals diese Aufmerksamkeit. Also, ich habe selber viel Source Code gelesen -- mußte ich, um das Linux-Handbuch zu schreiben -- und enorm viel gelernt, und ich bin dafür enorm dankbar. Ich lese es einfach nur, um es zu lernen, weil es mich bereichert."[197]

Die Forscherinnen des sozialwissenschaftlichen Projekts "Kulturraum Internet" am Wissenschaftszentrum Berlin fassen die Beweggründe der freien Entwickler folgendermaßen zusammen:

"An interesting question is what motivates people to contribute to a free software project. Individual and social motivations may be found:

- intellectual challenge and game
- creativity and pride of something self-made
- realizing something which fulfills one's demands on taste and quality
- socializing with people who share common ideas and interests
- fame
- fostering collective identity"[198]

Wenngleich die Motivation der freien Entwickler nicht auf wirtschaftlichen Profit zielt, zahlt sich die Möglichkeit, sich über ein Projekt zu profilieren, schließlich auch auf dem Arbeitsmarkt aus. Wer sich beim Management eines freien Projektes oder durch seinen Code bewiesen hat, ist ein begehrter Mitarbeiter der Software-Industrie. Im besten Fall wird er dann dafür bezahlt, im Rahmen seines Arbeitsverhältnisses weiterhin freie Software zu entwickeln. Cygnus z.B. ist eines der größten Unternehmen, das auf die Entwicklung und den Support von GNU-Software setzt. Auftraggeber sind häufig Hardware-Unternehmen, die die GNU-Tools (wie den C-Compiler) durch Cygnus auf eine neuen Architektur portieren lassen. Bezahlt wird die Arbeit, ihre Resultate stehen unter der GPL wieder allen zur Verfügung.[199] Auch der Unix-Distributor SCO, Linux-Kontor oder Be entwickeln freie Software stückchenweise im Kundenauftrag.

In zunehmendem Maße haben Leute also auch im Rahmen ihrer Anstellung bei einem Internet Service Provider, einem Hardware-Hersteller oder einer dedizierten Open-Source-Firma die Gelegenheit, an freier Software zu arbeiten. Eine Umfrage unter den XFree86-Entwicklern ergab, daß rund zehn Prozent von ihnen (65 Personen) ihren Lebensunterhalt direkt mit freier Software verdienen.[200] Linux-Distributoren wie RedHat, Caldera oder SuSE bezahlen Angestellte dafür, Software zu entwickeln, die z.T. an die Open-Source-Gemeinde freigegeben wird. Die Ausweitung der installierten Basis allein (der europäische Marktführer SuSE verkaufte von seinem Linux-Paket 6.0 innerhalb von zwei Monaten mehr als 100.000 Stück) gewährleistet Einnahmen aus Support, weiteren Dienstleistungen und kommerziellen Zusatzprodukten. Das kooperative Klima in einer solchen freien Umgebung motiviert auch

---

[197] Hetze, Wizards 7/1999 Diskussion

[198] Helmers/Seidler 1995

[199] http://www.redhat.com/about/cygnus_1999/redhat-cygnus111599.html; s.a. Tiemann 1999

[200] Hohndel, Wizards 7/1999-Diskussion

angestellte Entwickler, länger an einem Projekt zu arbeiten. Cygnus verzeichnet einen Personalwechsel, der nur ein Zehntel dessen anderer Silicon Valley-Firmen beträgt.[201]

Die Zusammensetzung der Entwicklergemeinschaft ist von Projekt zu Projekt sehr unterschiedlich. Auch nach der Art der Aufgaben unterscheiden sich die Beteiligten. Große Programmierarbeiten, wie Kernels, werden eher von festangestellten oder in ihrer Freizeit arbeitenden erfahrenen Programmierern übernommen, während die Entwicklung von kleineren Tools und das *Bugfixing* von einer viel größeren und heterogeneren Gruppe getragen wird.

> "FreeBSD kann sich den Luxus leisten, daß viele der Hauptentwickler festangestellt sind von größeren Firmen, z.B. von Walnut-Creek CD-ROM, die ja ihren FTP-Server-Betrieb auf FreeBSD fahren, wo FreeBSD das bestgehende CD-ROM-Produkt ist. Die haben zwei Leute angestellt, die nur FreeBSD entwickeln, d.h., die Firma sponsort praktisch die Open Source-Entwicklung. Matt Dillon, mittlerweile einer der Chef-Kernel-Entwickler, ist auch *Technical Director* bei einem der größten ISPs in Kalifornien, bei Best Internet. Und auch die erlauben ihm, damit er den Server-Betrieb am Laufen hält und es auch weiter skaliert, eben mitzuschreiben an den entsprechenden Kernel-Funktionen für *virtual Memory*, für SMP, für all das, was man braucht. Ein anderer sitzt bei der NASA und baut für die gerade einen Parallelrechner auf FreeBSD-Basis auf, und auch das geht alles als *Feedback* in das Projekt zurück. Soweit mal zum Kernel, wo wir den größten Teil an zentral anfallender Hauptarbeit haben. Bei den *Utilities* außenrum ist es wirklich ein Heer von Leuten, die Kleinigkeiten machen, so wie ich das hin und wieder auch tue."[202]

Mit der zunehmenden Kommerzialisierung der Open Source-Software nehmen auch die Bemühungen der Unternehmen zu, freie Entwickler anzuwerben. Bemerkenswert ist, daß viele den Verlockungen einer Festanstellung widerstehen. Linux-Entwickler Alan Cox sagte in einem c't-Interview, daß es viele seiner Kollegen vorzögen, ihre Entwicklungsarbeit weiterhin als Hobby zu betreiben. Statt weisungsgebunden zu arbeiten, schätzen sie die Freiheit höher, ihre eigenen Probleme zu wählen und sie in ihrer eigenen Zeit zu bearbeiten. Auch diejenigen, die mit Linux Geld verdienen, aber gleichzeitig unabhängig bleiben möchten, haben die Gelegenheit dazu. Cox: "Es gibt Hersteller, die ein Problem mit ihrer Hardware haben, aber nicht die Möglichkeit, es innerhalb ihrer Firma zu lösen. Solche Auftragsarbeiten werden weltweit vergeben - eine gute Sache für Leute, die nicht im Silicon Valley leben, sondern sonstwo in der Welt."[203]

Software-Zyklus: Entwickler, Power-User, Endnutzer

Man kann drei großen Phase im Leben einer freien Software unterscheiden. In der ersten Phase zieht es ausschließlich Entwickler an. Die Software ist noch nicht mehr als eine Rohskizze, ein Entwurf dessen, was es verspricht zu werden. Jeder kann sich die Software beschaffen, aber nur wer selbst am Quellcode mitarbeiten möchte, wird es tun.

---

[201] Tiemann 1999: 85

[202] Hausen, Wizards 7/1999-Diskussion

[203] Diedrich 19999

In der zweiten Phase hat die Software ein Stabilität erlangt, die sie für *Power-User* (z.B. Systemadministratoren) interessant macht, die sie nicht primär weiterentwickeln, sondern einsetzen wollen. Installation, Integration und Wartung erfordert auch jetzt noch weiterreichende Programmierfähigkeiten, doch diese Nutzergruppe erhält dafür eine Funktionalität, Flexibilität und ein Maß an Kontrolle über die Software, die sie zu einer attraktiven Alternative zu proprietärer Software macht.

Hat die Software einen Grad an Stabilität erreicht, die sie für den praktischen Einsatz geeignet macht, gibt das Projekt eine offizielle *Release* heraus. Parallel zu dieser Produktionsversion wird an einer Entwicklerversion weitergearbeitet. Diese Zweiteilung erlaubt einerseits eine kontinuierliche Verbesserung und Erweiterung, an der sich alle Interessierten beteiligen können, und andererseits das Einfrieren von stabilen Momentaufnahmen aus diesem Prozeß, die bis zur nächsten *Release* unverändert bleibt, für diejenigen, die die Software in ihrer täglichen Arbeit verwenden wollen.

In der dritten Phase erhält die Software Merkmale wie Installationshilfen und Dokumentation, die sie schließlich auch für technisch weniger bedarfte Nutzer zugänglich macht. Dieser Schritt wird oft nicht mehr von den selbstmotivierten Teilnehmern des freien Projekts durchgeführt -- *"Coders don't write docs"* --, sondern von Dienstleistungsfirmen, die oft eigens für den Support von freier Software gegründet wurden.

Auf dieser Stufe stellt sich auch die Frage einer Integration von Betriebssystem über die graphische Benutzeroberfläche bis zu den Anwendungen. Zentralistischen Modellen von Unternehmen wie Apple oder Microsoft gelingt dies naturgemäß gut, ebenso einem enger gekoppelten Projektverbund wie GNU. In einer weniger eng gekoppelten Umgebung besteht dagegen ein Bedarf an Abstimmung, ohne dafür jedoch auf zentralisierte Gremien zurück fallen zu wollen. Christian Köhntopp beschreibt die Problematik:

> "Wenn man Gesamtinstallationen eines Systems betrachtet, dann kann man sagen, die hat einen gewissen Reichtum oder eine Komplexität. Die ergibt sich einmal aus der Anzahl der Features, die in dem System zur Verfügung stehen und auf der anderen Seite aus der Anzahl der Integrations- oder Kombinationsmöglichkeiten dieser Features. Ein Problem, das ich bei vielen Projekten im OpenSource-Bereich sehe, ist eine schlechte oder mangelnde Koordination über solche Projektgrenzen hinweg. Viele solche OpenSource-Projekte schaffen es sehr gut, sich selbst zu managen und im Rahmen ihres Projektes auch entsprechende Integrationen zu schaffen, aber über Projektgrenzen hinweg ist das in der Regel deutlich schlechter entwickelt. Bei so großen Projekten wie KDE gibt es innerhalb des Projektes eine vergleichsweise gute Integration. Es war jedoch äußerst schmerzhaft, das KDE-Projekt und das vergleichbare Gnome-Projekt aufeinander abzustimmen. Es gab in der Vergangenheit sehr große Schwierigkeiten, etwa die XFree-Entwicklung und vergleichbare Projekte auf anderer Ebene miteinander abzustimmen. In diese Richtung müßten bessere Kommunikationsmöglichkeiten geschaffen oder unterstützt werden."[204]

Freie Software stößt auf gewisse Grenzen, da sie weiterhin Berührungsflächen mit einer unfreien Umgebung hat. So macht es proprietäre Hardware schwierig, Treiber für Linux und XFree86 zu schreiben. Jeder neue Drucker, jede Grafik- oder ISDN-Karte erfordert ein aufwendiges *Reverse Engineering*, um sie unter freier Software nutzbar zu machen. Mit der

[204] Köhntopp, Fachgespräch 7/1999

wachsenden Installationsbasis nimmt jedoch auch das Interesse von Hardware-Herstellern zu, ihre Produkte der Linux-Welt zugänglich zu machen, und sie bieten entweder von sich aus Treiber an oder stellen den Projekten unter Nichtweitergabebedingungen (NDA) die erforderliche Dokumentation zur Verfügung.

Nicht-freie Bibliotheken stellen eine weitere Bedrohung dar. Sie locken Programmierer mit attraktiven Features in die Falle der Unfreiheit. Die Problematik stellte sich zum ersten Mal in den Achtzigern mit dem proprietären Motif-Tookit, auf den das X Window System aufsetzte. Das freie XFree86 ersetzte Motif 1997 durch LessTif. Eine ähnliche Problematik ergab sich ab 1996, als der Grafik-Desktop KDE die proprietäre GUI Toolkit Library namens Qt der norwegischen Firma Troll Tech AS verwendete. Auch hier führte die Verwendung einer geschlossenen Software zu einem System mit mehr Fähigkeiten, aber weniger Freiheit. Drei Reaktionen darauf stellten sich ein. 1997 startete Miguel des Icaza ein neues Projekt für einen Grafik-Desktop namens GNOME (GNU Network Object Model Environment), das ausschließlich freie Software verwendet. Harmony ist eine Bibliothek, die Qt ersetzt und somit KDE zu einer vollständig freien Software macht. Schließlich veränderte Troll Tech auf Drängen der Linux-Welt 1998 die Lizenzbedingungen für Qt, so daß die Bibliothek seither in freien Umgebungen einsetzbar ist.[205]

Die Popularität und die inhärenten Vorteile des freien Entwicklungsmodells haben aus unterschiedlichen Gründen eine Reihe von Firmen veranlaßt, den Quellcode ihrer proprietären Software offenzulegen. Ob ein Software-Projekt erfolgreich von einem geschlossenen in einen offenen Modus umgeschaltet werden kann, ist jedoch fraglich. Der spektakulärste Versuch ist wiederum Netscapes Mozilla, das von vielen als gescheitert angesehen wird. Zunächst fand der offengelegte Quelltext eine enthusiastische Aufnahme. Bereits Stunden nach der Freigabe setzte ein Strom von Code ein, der jedoch bald versiegte. Die Hauptarbeit, die daran geleistet wurde, kam von Netscape-Angestellten.[206] Einer der Protagonisten, Jamie Zawinski, schrieb in seiner öffentlichen Kündigung: “For whatever reason, the project was not adopted by the outside. It remained a Netscape project... The truth is that, by virtue of the fact that the contributors to the Mozilla project included about a hundred full-time Netscape developers, and about thirty part-time outsiders, the project still belonged wholly to Netscape -- because only those who write the code truly control the project.”[207]

Im selben Text nennt Zawinski eine Reihe möglicher Gründe. Was Netscape freigab, war nicht der Code der aktuellen Version des Navigators, u.a. fehlten der Mailer ‘Communicator’, die Java- und die Kryptografie-Module. Es handelte sich um eine riesige Menge Code, aber nichts, was man tatsächlich hätte kompilieren und benutzen können. Die Menge war so groß, daß es sehr lange gedauert hätte sich einzuarbeiten, um sinnvolle Beiträge leisten zu können. Zu vermuten ist, daß auch die innere Struktur einer Software, die in einem geschlossenen Modus erarbeitet wird, eine andere ist, als die, die in einem freien Projekt entsteht. Selbst wenn der Quellcode vollständig einsehbar ist, können die

---

[205] Stallman 1999: 66 f.

[206] Ein halbes Jahr nach dem Start von mozilla.org bestand die Koordinationsgruppe aus drei Vollzeitbeschäftigten und einige Freiwilligen. Die meisten Entwickler (‘mehr als Hundert’) waren Netscape-Angestellte (Zawinski 11/1998). Im Juni 2000 besteht die Kerngruppe von mozilla.org aus 12 Leuten, die von ihren div. Arbeitgebern dafür bezahlt werde. Eine Statistik von Ende 1999 bis Mai 2000 zeigt pro Monat zwei bis drei Dutzend aktive Entwickler außerhalb von Netscape, etwa doppelt so viele Bug-Patches, und einige Hundert gemeldete Bugs, s. http://webtools.mozilla.org/miscstats/

[207] Zawinski 3/1999

grundlegenden Designentscheidungen hinter den Details der Codierung unsichtbar bleiben.[208] In seinem Fazit warnt Zawinski davor, aus dem Scheitern von Mozilla zu schließen, daß das Open Source-Modell insgesamt nicht funktioniere. “If there's a cautionary tale here, it is that you can't take a dying project, sprinkle it with the magic pixie dust of ‘open source,’ and have everything magically work out.”[209]

Eine letzte Frage, die sich in Bezug auf das freie Entwicklungsmodell stellt: kann es etwas grundsätzlich Neues hervorbringen? GNU/Linux, XFree86 oder GIMP sind von exisitierender proprietärer Software ausgegangen. Das bedeutet natürlich nicht, daß es einfach wäre, diese Programme unter freien Bedingungen zu rekonstruieren, aber die grundsätzlichen Designentscheidungen und die Zielvorgaben lagen vor, als die Projekte begannen. Es scheint, als könnte ein freies Projekt nur ein bekanntes Ziel verfolgen, aber sich keine neuen setzen. Ein Beleg dafür könnte der Betriebssystemskern des GNU-Projekts, der Hurd, sein, der dem in der Zeit aufregendsten, innovativsten Ansatz folgte, der Mikrokern-Architektur. Daß bis heute keine einsatzfähige Version des Hurd zustandegekommen ist, ist zweifellos auf dieselben Gründe zurückzuführen, daß sich Mikrokerne auch in der akademischen und kommerziellen Forschung nicht haben durchsetzen können. Torvalds dagegen wählte bei seinem Ansatz für Linux das altbewährte Makrokernmodell (wofür er von Andrew Tanenbaum öffentliche Schelte bezog). Raymond fragt: “Suppose Linus Torvalds had been trying to pull off fundamental innovations in operating system design during the development; does it seem at all likely that the resulting kernel would be as stable and successful as what we have?” Er behauptet kategorisch “One can test, debug and improve in bazaar style, but it would be very hard to originate a project in bazaar mode.”[210] Bei einem Betriebssystem, also einer Infrastruktur für Anwendungs-Software, ist es wichtiger, daß es stabil und zuverlässig ist, als daß es radikal neue Strukturen ausbildet, was auch zur Folge hätte, daß alle darüber laufende Anwendungs-Software umgeschrieben werden müßte. Wenn auch noch zu beweisen wäre, daß auch freie Projekte zu grundlegenden, revolutionären konzeptionellen Sprüngen in der Lage sind, steht außer Zweifel, daß sie sich für eine evolutive Weiterentwicklung in kleinen iterativen Schritten hervorragend eignen. In einigen Fällen hat sich der Fokus der Innovation von den proprietären Vorlagen zu den freien Projekten verlagert.

> “War das PC-X anfangs ein Anhängsel der großen, seriösen Workstation-X-Entwicklung, hat sich das Verhältnis inzwischen umgekehrt. Auch Jim Gettys, einer der Urväter von X, sieht heute XFree86 als die Gruppe, die die Entwicklung von X an sich weiter trägt. X.Org soll diese Rolle wieder übernehmen, sofern die Gruppe endlich mal ins Rollen kommt, aber im Moment ist es die XFree86-Gruppe.”[211]

---

[208] “Complexity and size effectively close source code for system programming projects like OSes compilers after, say, 100K lines of code without good higher level documentation or participation in he project from its early stages... ‘Reveal the code, conceal the kitchen’.” Bezroukov 12/1999

[209] Zawinski 3/1999

[210] Raymond 1998: Chapter 9

[211] Hohndel, Wizards 7/1999

Die Software

Die Zahl und die funktionale Bandbreite der freien Programme ist unüberschaubar. Ein großer Teil hat ‘infrastrukturellen’ Charakter. Besonders die Bereiche Internet, Betriebssysteme (Emulatoren) und Entwickler-Tools ziehen die Aufmerksamkeit freier Entwickler auf sich. Schon grafische Desktop-Umgebungen (KDE, Gnome), die man ebenfalls als informatische Infrastruktur ansehen kann, sind vergleichsweise jung. Viele Applikationen stammen aus Universitäten und hier vor allem aus dem naturwissenschaftlich-technischen Bereich. Insgesamt läßt sich sagen, daß für alle Einsatzbereiche des Computers, einschließlich Büroanwendungen, Spiele, bis zu Videoschnittsystemen oder 3D-Modellierung freie Software exisitiert. Wer sich eine der Linux-Distributionen beschafft, erhält damit bereits zahlreiche stabile Softwarepakete (in der aktuellen SuSE Linux 6.4 z.B. mehr als 1.500)

Freie Software ist (ebenso wie proprietäre Software) nirgends katalogisiert. Ein großes Portal für Linux Software, das *Dave Central* von Dave Franklin,[212] verweist auf mehr als 4.000 Programme. Die größten Kategorien sind Netzwerkprogramme, Programmierung, Systemwerkzeuge und Büroanwendungen. Handelt es sich dabei überwiegend um stabile, direkt anwendbare Programme, so bietet *SourceForge*[213] eine Einblick in die aktuellen Entwicklungsaktivitäten. SourceForge ist ein Angebot von VA Linux Systems, Inc.[214], das Open-Source-Entwicklern Hardware- und Software-Ressourcen wie CVS-Repositorium, Mailinglisten, Bug-Tracking, Dateiarchive, Foren und eine Web-basierte Administration zur Vergügung stellt. Dort sind (im Juni 2000) 3.500 Projekte gelistet, zwei Drittel davon auf Linux (rsp. POSIX), ein knappes Drittel auf X11, ein Fünftel mit stabilen Produktionsversionen. Ein Drittel wird in C erstellt, ein weiteres knappes Drittel in C++, die übrigen in div. Programmiersprachen (Perl, Java, PHP, Python, Assembler). Bis auf wenige Ausnahmen stehen die Projekte unter einer von der *Open Source Initiative* gutgeheißenen Lizenz. Die Hälfte der Software richtet sich an Entwickler, die andere Hälfte an Endnutzer. Bei den Anwendungsbereichen bilden Internet, Kommunikation, System- und Entwicklungssoftware, Multimedia und Spiele die größten Gruppen. Eine weiterer wichtiger Startpunkt für die Suche nach bestimmten GNU/Linux-Programmen und für tägliche Updates über neue Projekte und neue Versionen ist Freshmeat[215]

Im Folgenden werden Kurzbeschreibungen und Verweise auf einige ausgesuchte freie Software-Projekte geben, gefolgt von längeren Ausführungen zu BSD-Unix, der Debian GNU/Linux-Distribution, XFree86, KDE, Apache und GIMP.

- GNU-Software[216] -- mehr als 200 im Rahmen des GNU-Projekts erstellte Programme[217] von den Binutils über ein CAD-Programm für Schaltkreise bis zu

---

[212] http://linux.davecentral.com/

[213] https://sourceforge.net/

[214] eine Firma, die Linux-Systeme und Support verkauft. Ihre Motivation für die Investitionen in SourceForge: “If the selection and quality of Open Source software improves, VA can offer its customers more competitive solutions.” (https://sourceforge.net/docs/site/faq.php)

[215] http://freshmeat.net/

[216] http://www.gnu.org/software/software.html

[217] sehr viel mehr Programme stehen unter der GNU-Lizenz, sind aber nicht innerhalb des GNU-Projekts entstanden.

Spielen, darunter solche Standards wie Bash (die Bourne Again SHell), CVS, Emacs, Ghostscript und Ghostview für die Erzeugung und Darstellung von PostScript- und PDF-Dateien, der Dateimanager Midnight Commander, das Notensatzprogramm lilypond, der Webseiten-Downsloader wget, ein GNU-Java-Top-level-Paket mit div. Anwendungen und GNOME.

- GNOME (GNU Network Object Model Environment)[218], eine graphische Desktop-Umgebung. Das Projekt wurde Anfang 1998 gegründet. Es verwendet das aus dem GIMP-Projekt (s.u.) hervorgegangene Toolkit Gtk und umfaßt viele Werkzeuge für die tägliche Arbeit, wie Datei-Manager und kleine Office-Applikationen, insgesamt über 230 Programme.

- BIND (Berkeley Internet Name Daemon)[219] ist der de facto Standard-DNS-Server für das Internet. Das Domain Name System des Internet wurde anhand von BIND entwickelt. BIND wird ebenso wie DHCP (eine Implementation des Dynamic Host Configuration Protocol) und INN (das InterNetNews-Package, ursprünglich geschrieben von Rich Salz) heute vom Internet Software Consortium[220] betreut.

- Sendmail[221] wickelt den Transport von 90% des weltweiten eMail-Verkehrs ab. Es stammt aus der Berkeley Universität. Sein Autor, Eric Allman gründete 1997 eine Firma[222] darauf, doch parallel dazu ist die quelloffene, freie Version weiterhin verfügbar.

- Squid[223] ist ein Proxy Server, der auf dem ICP Protokoll basiert. Squid ist populär bei großen ISPs.

- SAMBA[224] -- ein SMB-File- und Druck-Server für Unix. Vor kurzem hat es die SAMBA Mannschaft geschafft, auch einen NT-Controller für Unix zu entwickeln. Seit der Version 2.0 werden auch MS-Windows-Clients unterstützt. SGI engagiert sich ebenfalls stark in SAMBA.

- PERL (Practical Evaluation and Reporting Language)[225] ist die Standard-Scriptsprache für Apache-Webserver (s.u.). PERL ist auf Unix sehr beliebt, vor allem aufgrund seiner leistungsstarken Text/Zeichenkettenverarbeitung und des Vertrauens auf Befehlszeilenverwaltung aller Funktionen. Die umfangreichste Bibliothek mit freien Perl Skripts ist CPAN.[226]

---

[218] http://www.gnome.org
[219] http://www.bind.org und http://www.isc.org/bind.html
[220] http://www.isc.org/
[221] http://www.sendmail.org
[222] http://www.sendmail.com
[223] http://squid.nlanr.net
[224] http://www.samba.org
[225] http://www.perl.org
[226] http://cpan.org

- Ghostscript[227] ein PostScript-Interpreter, geschrieben von L. Peter Deutsch, unter GPL entwickelt, dann auf die Aladdin-Lizenz umgeschwenkt.[228]

- Majordomo[229] der vorherrschende Mailinglisten-Server im Internet ist in PERL geschrieben und ebenfalls ein OSS Projekt.

- WINE (Wine Is Not an Emulator)[230] ist ein Windows Emulationsbibliothek für Unix.

- Zope (Z Object Publishing Environment)[231] eine Web-Applikationsplattform für die Generierung von dynamischen Webseiten. Basiert auf der Skriptsprache Python und auf Produkten der Firma Digital Creations, die auf Empfehlung eines Venture-Capital-Investors 1998 unter eine Open-Source-Lizenz[232] gestellt wurden.

- OpenBIOS[233] ein Projekt, eine IEEE 1275-1994-Standard Firmware zu schaffen, die proprietäre PC-BIOSe ersetzen soll. Die erste Entwicklerversion 0.0.1 wurde im November 1998 freigegeben.

BSD

1977 hatte Bill Joy an der Berkeley University die erste “Berkeley Software Distribution” (BSD) mit Unix-Programmen und im Jahr darauf die vollständige Unix-Distribution 2.11BSD zusammengestellt. Die Version 4.2BSD (1983), die ein schnelles Dateisystem und TCP/IP enthielt, wurde auch unter kommerziellen Anbietern von Unix-Rechnern populär und trug viel zur weltweiten Verbreitung von Unix und Internetzwerken bei. Während sich die Anbieter kommerzieller Unix-Versionen gegenseitig die Lizenzdaumenschrauben anlegten, gab die Berkeley-Gruppe 1989 zunächst die Netzwerkelemente aus BSD als Networking Release 1 unter der eigens dafür entwickelten freien BSD-Lizenz heraus. In einer im Vergleich zum Linux-Projekt wenig beachteten offenen, Internet-basierten Entwicklung wurden daraufhin alle geschützten Unix-Module durch freien Code ersetzt, der 1991 als Networking Release 2 erschien. Diese Version portierte Bill Jolitz auf den PC und veröffentlichte sie Anfang 1992 als 386/BSD unter der BSD-Lizenz. Um das 386/BSD sammelten sich enthusiastische Nutzer und Entwickler, die es als NetBSD-Gruppe weiter pflegten. Damit migrierte das Projekt vollends aus der Akademie ins Internet. Die Forschungsgruppe an der Berkeley Universität legte 1995 letzten Änderungen als 4.4BSD-Lite, Release 2 vor und löste sich dann auf.[234]

---

[227] http://www.ghostscript.com

[228] Interview mit Deutsch zur Geschichte des Projekts, Lizenzen und Zukunft: http://www.devlinux.org/ghost/interview.html

[229] http://www.greatcircle.com/majordomo

[230] http://www.wine.org

[231] http://www.zope.org/

[232] http://www.zope.org/Resources/License

[233] http://www.freiburg.linux.de/OpenBIOS/

[234] ausführlicher s.o. unter “Geschichte: Unix” und McKusick 1999: 33 f.

Gleich zu Anfang der Weiterentwicklung des 386/BSD kam es, aus weitgehend persönlichen Gründen, zu einer Spaltung. Die **NetBSD**-Gruppe[235] zielte darauf, eine möglichst große Verbreitung auf möglichst viel Plattformen zu erreichen, mit dem Ergebnis, daß es heute fast keinen 32-Bit-Prozesser mit einer Memory Management Unit gibt, auf dem NetBSD nicht läuft. Das zweite daraus hervorgegangene Projekt **FreeBSD**[236] zielte darauf, ein leistungsfähiges, stabiles Unix-Server-Betriebssystem für Intel-CPUs zu schaffen. FreeBSD ist auch auf den Alpha portiert, aber der Schwerpunkt liegt auf Intel. FreeBSD hat heute die größte Installationsbasis der aus dem Net Release 2 abgeleiteten Systeme. 1997 kam es innerhalb der NetBSD-Gruppe zu einer weiter Spaltung. Daraus ging ein Projekt namens **OpenBSD**[237] hervor, das das Betriebssystem auf Sicherheit hin optimiert. Die unterstützen Hardwarplattformen sind ähnlich wie bei NetBSD.

Die drei Projekte bestehen freundschaftlich nebeneinander und übernehmen Innovationen voneinander. Diese Zuammenarbeit gibt es auch mit der Welt des parallel heranreifenden Linux und von anderen Betriebssystemen,[238] mit XFree86, Apache, KDE und weiteren Projekten. Der Source-Baum liegt unter CVS-Kontrolle. Man kann lesend und schreibend weltweit auf ihn zugreifen.

Während eine Linux-Distribution um den Kernel herum Software-Pakete aus den unterschiedlichsten Quellen teilweise in Binärform zusammenträgt, für die verschiedene Maintainer zuständig sind, gibt es bei BSD ein zentrales Source-Code-Management für den Kernel *und* alle Utilities (die GNU-Tools, wie der GCC, UUCP, TOP usw.). Installiert wird aus dem Quellcode. Installierer wie der RedHat Package-Manager (rpm) oder dselect, die fertige Binary-Packages installieren, wie das in der PC-Welt üblich ist, sind in der Unix-Welt eine relativ junge Erfindung. Zur Automatisierung der Source Code-Installatation dient bei FreeBSD die *Ports Collection*. Sie umfaßt heute einen Source-Baum mit *Make Files* für mehrere tausend Applikationen und einem Umfang von 10 bis 15 MByte. Diesen Baum spielt sich der Nutzer auf seine lokale Festplatte und wählt die *Make*-Dateien der Software-Pakete aus, die er installieren möchte. *Make* ist ein Projektverwaltungs-Tool, mit dem man automatisiert Software übersetzen kann. Werden die entsprechenden *Make*-Dateien aufgerufen, so werden die Pakete für den Apache, KDE usw. direkt vom Original-FTP-Server geholt, eventuell für FreeBSD angepaßt, Variablen werden gesetzt, der Quellcode durchkompiliert, installiert und in einer Package-Datenbank registriert, so daß sich alles wieder sauber deinstallieren läßt. Bei der Erstinstallation erzeugt der Befehl *make world* auf diese Weise das gesamte BSD-System. Updates erfordern nur, daß ein Eintrag im entsprechenden *Make File* geändert wird, und sofort werden die aktuellen Sourcen z.B. des Apache über das Internet auf den eigenen Rechner übertragen. Die *Ports Collection* erlaubt es auch, vollautomatisiert in bestimmten Abständen, z.B. jede Nacht, Updates zu holen. Ein Systemadministrator kann das Update auf einem Rechner durchführen und dann per NFS auf alle BSD-Rechner in seinem Netz exportieren, was der Verwaltung einer größeren Menge von Servern sehr förderlich ist.

---

[235] http://www.netbsd.org/

[236] http://www.freebsd.org/

[237] http://www.openbsd.org/

[238] "Der OS/2 TCP/IP-Stack z.B. ist 100% BSD. Wenn Sie sich das Net-Stat-Kommando auf einem OS/2-Rechner angucken, dann stellen sie fest, daß da Ausgabe von internen Strukturen möglich ist, wie sie nur im BSD TCP/IP-Stack existieren." (Hausen, Wizards 7/1999)

Sämtliche Open Source-Projekte, die man aus der Linux-Welt kennt, laufen ebenfalls auf den drei BSD-Varianten. Neben mehreren Tausend freien Applikationen lassen sich auch proprietäre *binary-only* Applikationen, wie z.B. WordPerfect oder eCommerce-Software, unter BSD betreiben.

Die BSD-Lizenz[239] ist gemäßigter als die GNU-GPL. Die Namen der Autoren und der Universität Berkeley muß auch in abgeleiteter Software genannt bleiben. Zugleich darf nicht mit dem Namen der Universität für das neue Produkt geworben werden. Abgeleitete Closed Source-Produkte sind erlaubt.

Unter den Benutzern von FreeBSD finden sich so illustre Namen wie Yahoo und Hotmail. Microsoft versucht seit dem Ankauf von Hotmail erfolglos, diesen kostenlosen eMail-Service auf NT-Server umzustellen. Der größte FTP-Server der Welt, mit 5-6.000 Nutzern gleichzeitig und einem Terra-Bit an täglichem Datendurchsatz ist ein FreeBSD-Rechner.[240]

Debian GNU/Linux

Debian[241] startet 1993, als Ian Murdock im Usenet zur Mitarbeit an einer neuen Linux-Distribution aufrief. Die Distributionen der Zeit litten noch an Kinderkrankheiten. Abhängigkeiten und Konflikte wurden nicht berücksichtigt. Zu Binärpaketen fand sich kein Quellcode. Fehler wurden in neue Versionen übertragen, obwohl bereits Korrekturen vorlagen. Die Lizenzen von Komponenten ließen sich nicht ersehen. Hier wollte Murdock und ein Dutzend Mitstreiter Abhilfe schaffen.

Bei Debian wie bei den anderen Distributionen steht nicht die Entwicklung von Software, sondern ihre Integration in ein stabiles Gesamtsystem im Vordergrund. Zentrales Instrument, um aus einer Anhäufung von Einzelelementen ein System zu komponieren, ist die Paketverwaltung. Sie protokolliert bei der Installation, welche Dateien zu einem Paket gehören, so daß sie sauber deinstalliert oder von neuen Versionen überschrieben werden können. Sie verhindert, daß konfligierende Programme installiert werden und daß Programme, die von anderen abhängen, nur mit diesen zusammen installiert werden (ein *cron*-Paket z.B. benötigt einen Mailserver, da es bei Fehlern eMails verschickt). Sie verwaltet Konfigurationsdateien, integriert Hilfedateien ins System und macht *Shared Libraries* dem System bekannt. Das Debian-Projekt erstellte also als erstes den Paketmanager *dpkg*. Weitere Programme (*dselect, apt*) holen die gewünschten Programme von FTP-Server oder CD, untersuchen sie auf Abhängigkeiten und Konflikte und übergeben sie dann an *dpkg*. Mit dem Programm *alien* lassen sich auch Pakete anderer Distributionen, wie RedHat oder Slackware, auf Debian installieren und umgekehrt.

Die Arbeit begann mit finanzieller Unterstützung der FSF. In den ersten drei Jahren baute Murdock das Projekt auf. Seither wird der Projektleiter jährlich aus den Reihen der Mitarbeiter gewählt. Es waren dies Bruce Perens, Ian Jackson, und seit 1999 Wichert Akkerman. Der Projektleiter übergibt die Verantwortung für die Betreuung einzelner Pakete, für Dokumentation, Lizenz- und Policy-Fragen, Webseiten, Mailingslisten usw. an andere

---

[239] http://www.freebsd.org/copyright/license.html

[240] Patrick M. Hausen, Wizards 7/1999

[241] http://www.debian.org, der Name setzt sich aus Debra und Ian Murdock zusammen, die das Projekt gestartet haben.

Freiwillige. Das Debian-Team besteht aus weltweit etwa 500 Mitarbeitern im Alter von 13 bis 70 Jahren. Das Projekt ist offen für alle. Wer Lust hat, mitzuarbeiten, meldet sich einfach beim Debian-Team. Ein Paketbetreuer pflegt Pakete, die er selbst benutzt und gut kennt, und an deren Qualität er daher ein persönliches Interesse hat. Er verfolgt die Entwicklung in den jeweiligen Software-Projekten, wählt neue Versionen und neue Software aus, die in die Debian-Distribution aufgenommen werden, stellt sie zusammen, überprüft ihre Lizenzen daraufhin, ob die Programme in Debian aufgenommen werden können und spielt den Vermittler zwischen den Debian-Nutzern und den eigentlichen Software-Entwicklunsprojekten.

Die Pakete werden im Binärform und in einem Parallelverzeichnis im Quellcode in den Software-Baum eingefügt. Vor einer neuen Release wird ein *Code-Freeze* gemacht, d.h. der Software-Baum wird für Neueinträge gesperrt, der vorliegende Code wird debugged, was etwa zwei Monate dauert, und schließlich freigegeben. Das Debian-Team erstellt das *Master-Image* einer kompletten CD mit einem Umfang von 640 MByte und stellt es auf dem FTP-Server bereit. Im Interesse der Stabilität werden neue Versionen erst releast, wenn die Pakete die Richtlinien für die Organisation des Systems (die *Policy*[242]) erfüllen und alle kritischen Bugs behoben sind. Das Bug-Tracking-System beruht auf einem mehrstufigen eMail-Austausch, der im Web-Archiv[243] von jedem eingesehen werden kann. Neben den Mailinglisten verwendet die Debian-*Community* auch den *Internet Relay Chat* (IRC) als Kommunikationskanal.

Anders als bei anderen Linux-Distributionen steht hinter Debian keine Firma, die Werbung und Vetrieb übernimmt. Debian ist eine Projektgruppe unter dem Schirm der *Software in the Public Interest* (SPI), einer gemeinnützigen Firma, die dafür gegründet wurde, Spenden entgegenzunehmen, und die Domain und das Warenzeichen von Debian und anderen Projekten anzumelden.

Debian GNU/Linux ist die ‘freieste’, dem GNU-Projekt am nahesten stehende Linux-Distribution. Änlich wie das GNU-Projekt und im Gegensatz zu den anderen Linux-Distributionen ist Debian sehr darauf bedacht, ausschließlich Software unter freien Lizenzen aufzunehmen (z.B. schließt Debian aufgrund der problematischen Lizenz der Qt-Bibliothek die graphische Desktop-Umgebung KDE von seiner Distribution aus.[244]). Dafür ist ein Kriterienkatalog (die *Debian Free Software Guidelines*) erarbeitet worden, an dem Lizenzen überprüft werden (Unbeschränkte Weitergabe, Verfügbarkeit des Quellcodes, Modifikationsfreiheit, keine Diskriminierung von Personen und Gruppen, keine Diskriminierung von Einsatzbereichen usw.). Statt einer Lizenz regelt ein ‘Gesellschaftsvertrag’ (den *Debian Social Contract*[245]) das Verhältnis unter allen an Debian Beteiligten. Darin heißt es, daß Debian 100% freie Software bleiben wird, daß neue Komponenten als freie Software der Gemeinschaft zur Verfügung gestellt wird, daß Probleme nicht verborgen werden und daß den Anwender und der Gemeinschaft freier Software oberste Priorität zukommt. Da zugestanden wird, daß einige Debian-Anwender Programme einsetzen müssen, die nicht den Kriterien für freie Software genügen, wurde für

---

[242] in welchen Verzeichnissen Konfigurationsdateien, ausführbare Programme, Dokumentation usw. abgelegt werden, wie Pakete benannt werden, welche IDs für Subsysteme verwendet werden dürfen usw. s. Debian Policy Manual, ftp://ftp.debian.de/pub/debian/doc/package-developer/policy.text.gz

[243] http://bugs.debian.org

[244] Carter 6/2000

[245] http://www.debian.org/social_contract

solche Programme ein eigener Bereich *non-free* auf dem FTP-Archiv eingerichtet. Sie sind nicht Bestandteil des Debian-Systems (*main*), werden aber dennoch für eine Zusammenarbeit mit ihm aufbereitet und in der Bug-Datenbank und den Mailinglisten mitbehandelt. Das Debian-Team bemüht sich darum, daß die Lizenzen von solchen Programmen angepaßt werden und erzielte dabei auch schon einige Erfolge. Ein weiterer Bereich *non-US* enthält Kryptografie-Software, die in den USA besonderen Auflagen unterliegt. Zu den weiteren Projekten gehören Debian GNU/Hurd, das die gleichen Pakete, aber an Stelle von Linux als Kernel den GNU-Hurd verwendet und das Debian-Beowolf-Projekt, bei dem viele Rechner zu einem Cluster zusammengeschaltet werden. Aus Debian sind russische, französische, italienische und japanischen Distributionen hervorgegangen, und auch Corel hat seine Linux-Distribution auf Basis von Debian erstellt.

Debian GNU/Linux ist die Linux-Distribution mit der größten Anzahl von Binärpaketen (in der Version 2.1 "Slink"[246] waren es 1.500 Pakete, in der Version 2.2 "Potato" vom Frühjahr 2000 schon doppelt so viele[247]), der größten Anzahl unterstützter Architekturen (Intel, Alpha, 68000, Power-PC, Sparc, Arm, Amiga, Atari, RS/6000, UltraSparc), der größten Anzahl Mitarbeiter und der längsten Testphase vor einem Release.[248]

XFree86

Das X-Window-System wurde 1984 von Jim Gettys und anderen am MIT entwickelt und ist heute in der Linux- und Unix-Welt der Standard für die Basis von graphischen Benutzeroberflächen. Noch am MIT entstand unter Industriebeteiligung zu seiner Implementierung und Weiterentwicklung das X-Consortium, das 1986 die erste kommerzielle Version vorlegte. 1993 endete das universitäre Engagement, und die Technologie ging an das neugegründete X Consortium, Inc. mit weltweit über 60 Mitgliedsunternehmen über. Vier Jahre später übertrug dieses Gremium die Verantwortung für X-Window an das Open Group-Konsortium,[249] das ein breiteres Spektrum von Technologien standardisiert, testet und zertifiziert.

Als 1992 das MIT-X-Consortium die Version X11R5 veröffentlichte, war zum ersten Mal auch ein X-Server für PCs dabei: X386. Da er nicht sehr schnell und stabil war, begann eine kleine Gruppe sogleich, ihn weiterzuentwickeln. Am Anfang waren es vier Entwickler, die einige kleine Patches austauschten, und ein paar Tester. Die erste Release, die daraus folgte, hieß X386 1.2.E (E, wie erweitert). Da Linux einen wachsenden Bedarf nach einer Implementation des X-Windows-Systems für PC-basierte Unix-artige Systeme schuf, entstand daraus ein eigenständiges Projekt, das sich mit einem Wortspiel auf X*Three*86

---

[246] Jede Release trägt zusätzlich zur Versionsnummer den Namen einer Figur aus dem Film "Toy Story", was seinen Grund vermutlich darin hat, daß Bruce Perens bei den Pixar Animation Studios von Steve Jobs arbeitet, die "Toy Story" produziert haben.

[247] Software aus dem GNU-Projekt wie GCC, XFree86, Gnome, g-libc, verschiedenen Window-Manager (FVWM, FVWM2, FVWM 95), Entwicklerwerkzeuge, Perl, Libraries, Textverarbeitungsprogramme wie AbiWord, Zeichentools wie GIMP, Datenbanken wie MSQL und MySQL, Internet-Applikationen wie Web-Server und Browser, Gnumeric aus dem Gnome-Paket, der WindowMaker Enlightenment usw.

[248] Schulze 3/1999 und Frank Ronneburg, Wizards 7/1999

[249] http://www.opengroup.org/. Presseerklärung "X Consortium to Transfer X Window System(tm) to The Open Group" Cambridge, Massachusetts, July 1, 1996, http://www.opennc.org/tech/desktop/Press_Releases/xccloses.htm

“XFree86”[250] nannte. Linux und XFree86 haben sich gegenseitig vorangetrieben. Linux ist weitergekommen, weil es eine graphische Oberfläche gab, und XFree86 ist weitergekommen, weil es mit Linux eine frei verfügbare Plattform dafür gab.

Die XFree86-Gruppe hat seither etwa zwei bis vier Releases im Jahr herausgebracht. Im Juli 1999 erschienen zwei neue Versionen, die praktisch einsetzbare 3.3.4., die die 3.3.3.1 ersetzt, sowie die experimentelle 3.9.15, die auf ein fundamental neues Design zielt, das in XFree86 4.0 münden wird. Letztere ist noch nicht für den praktischen Einsatz geeignet, bietet aber einen Einblick in die laufenden Entwicklungspläne.

XFree86, das heute etwa zweieinhalb Millionen Code-Zeilen umfaßt, läuft auf einer breiten Palette von Prozessoren (Intel, Alpha, Sparc, Power-PC, dem Itzy, einem Handheld von Compaq, usw.) und Betriebssystemen (Linux, Realtime-Betriebssysteme wie QNX, OS/2, Minix usw.). Die Zahl der XFree86-Nutzer wird konservativ auf 12 bis 14 Millionen geschätzt.

Das *Core-Team* des Projekts besteht aus elf Personen, die die Entwicklung steuern und zu einem großen Teil selbst vornehmen, die Source-Verwaltung mit Hilfe von CVS machen und sich darum kümmern, was an neuen Funktionen aufgenommen wird. Mit etwa 600 Entwicklern weltweit ist XFree eines der größten freien Projekte.

XFree86 verwendet eine Lizenz,[251] die auf die ursprüngliche MIT-X-Window-Lizenz zurückgeht, die wiederum von der BSD-Lizenz abgeleitet, aber noch schwächer als diese ist. Sie erlaubt Verwendung, Verbreitung, Modifikation und Verkauf unter der einzigen Bedingung, daß der Copyright-Vermerk erhalten bleibt. Sie verzichtet insbesondere auf die Auflage (z.B. in der BSD-Lizenz), dieselben Freiheiten auch für abgeleitete Software zu gewährleisten und (z.B. in der GNU-GPL) den freien Code nicht in proprietären zu integrieren. Diese Lizenz ermöglichte einerseits, daß viele kommerzielle Unix-Systeme heute X-Server enthalten, die auf XFree86 basieren. Andererseits führte sie zu einer der traurigsten Episoden in der Geschichte der freien Software. Im April 1998 kündigte die Open Group eine geänderte Lizenzpolitik für X Window an. Die neue Version X11R6.4 wurde proprietär und nur für zahlende Kunden verfügbar.[252] Sogleich hielt die Open Group ausgerechnet einige Sicherheits-Bugfixes zurück. Für XFree86 wird damit eine Zusammenarbeit unmöglich. Es geht seinen eigenen Weg. Im September 1998 sieht sich die Open Group gezwungen, die Entscheidung zurückzunehmen und X11R6.4 wieder unter einer quelloffenen Lizenz freizugeben. Hintergrund war neben dem Protest vor allem, daß sich der Fokus der Entwicklung zu XFree86 verlagert hatte, und innerhalb des Konsortiums kaum noch jemand daran arbeitete. War das PC-X anfangs ein Anhängsel der ‘großen, seriösen’ Workstation-X-Entwicklung, hatte sich das Verhältnis inzwischen umgekehrt. Auch Jim Gettys, einer der Urväter von X, sieht heute XFree86 als die Gruppe, die die Entwicklung von X an sich weiter trägt.

Nach der Verzweigung gab es zwischen dem industriegestützten X-Window und dem freien XFree86 weiterhin Zusammenarbeiten, die jedoch an der zentralen Frage des geistigen Eigentums immer wieder zu Konflikten führte. Das Konsortiums-X-Window setzt auf das proprietäre Tookit *Motif* auf, und auch die darüberliegende Desktop-Umgebung CDE (*Common Desktop Environment*) ist nicht quelloffen. Das freie XFree86 ‘erbte’ diesen

---

[250] http://www.XFree86.org/

[251] XFree86 License, http://www.xfree86.org/4.0/LICENSE1.html

[252] Richard Stallman sieht darin, daß diese Schließung möglich war, eine Bestätigung dafür, daß die X-Lizenz für die Zwecke freier Software mangelhaft ist. (Stallman 1998)

integrierten Technologie-Stack aus X, Motif und CDE, was in der offenen Weiterentwicklung immer wieder zu Schwierigkeiten führte. Daher ersetzte es Motif 1997 durch LessTif, und als quelloffene Alternative zu CDE entstand KDE (s.u.). In dem freien Projekt GIMP (s.u.) entstand als weitere Alternative zu Motif die Toolkit-Library Gtk, auf der der Desktop Gnome beruht.

Im Mai 1999 gründete sich aus der Open Group die non-profit Organisation X.Org[253] als offizieller Verwalter der X Window System-Technologie. X.Org unterhält den technischen Standard und veröffentlicht (kostenlos) die offiziellen Beispielimplementationen. Bereits 1994 hatte die XFree-Gruppe eine *Non-Profit Corporation* gegründet, da nur Unternehmen Mitglied des X-Consortiums werden können. Das Konzept einer losen Gruppe von Entwicklern macht es schwierig, Kooperationen oder Verträge mit einzelnen Firmen oder Industriekonsortien abzuschließen. *Non-profit* bedeutet, daß sich das Untenehmen aus Spenden finanziert, die für Mitgliedschaften in Konsortien und Infrastruktur wie Hardware und Internet-Access verwendet werden. Das Unternehmen hat keine bezahlten Mitarbeiter. Im November 1999 wurde XFree86 dann zum Ehrenmitglied von X.Org

XFree86 ist ein Open Source-Projekt. Die Releases werden zusammen mit dem Quelltext freigegeben. Es ist wiederum die Zusammenarbeit mit Firmen, besonders den Herstellern von Grafikkarten, die das Projekt zu einem teilweise geschlossenen Entwicklungsmodell zwingt. Diese Firmen hüten ihre Hardware-Dokumentationen und geben sie meist nur unter einem *Non-Disclosure Agreement* heraus, wonach der Source-Code veröffentlicht werden darf, aber nicht die Dokumentation. Damit das XFree-Projekt seinen Entwicklern Zugang zu diesen Dokumentationen geben kann, müssen diese förmliche Mitglieder (*Non-Voting Members*) werden und sich zur Nichtweitergabe verpflichten. Die Freiheit wird dadurch gewährleistet, daß jeder Mitglied werden kann. Dennoch ergeben sich daraus zwei Klassen von Source Code: der *Release-Code*, der für jeden frei verfügbar ist, und die *Internal Development Sources*, die nur den offiziellen Mitgliedern zugänglich sind.

Core-Team-Mitglied Dirk Hohndel gesteht die Bedenken über die fehlende Absicherung der Freiheiten in der XFree86-Lizenz zu, die dazu führt, daß jemand ein kommerzielles Produkt daraus machen und die freie Software-Welt ausschließen kann. Tatsächlich sei XFree86 weiterhin der einzige X-Server für Linux. Die wenigen Anbieter proprietärer Produkte hätten im Markt nicht Fuß fassen können.

> "Das zeigt, daß man in einem Open Source-Projekt rein mit der Qualität des Produkts, das man abliefert, dafür sorgen kann, daß für jemanden, der versucht, hier Closed Source zu gehen und daraus ein proprietäres Produkt zu machen, überhaupt kein Platz ist. Denn jeder der heute hingeht und dieses Projekt zumachen und daraus seine eigene Sache machen will, muß damit rechnen, daß wir all das, was er anbietet, auch anbieten. Wir bieten es mit Sourcen an und frei verfügbar."[254]

KDE

[253] http://www.x.org; zu den Mitgliedern gehören Compaq, HP, IBM, Silicon Graphics, Sun, Siemens und als Ehrenmitglied eben XFree86

[254] Hohndel, Wizards 7/1999

Die integrierten graphischen Arbeitsoberflächen von Desktop-Betriebssystemen wie MS-Windows oder Mac-OS haben wesentlich zur Popularisierung des PCs beigetragen. Unter Unix sind verschiedene Entwicklungen in diese Richtung unternommen worden. Eine davon, das *Common Desktop Environment* (CDE), hielt zusammen mit XFree86 und Motif Einzug in verschiedene Linux-Distributionen. Da CDE jedoch proprietär ist, kann es nicht nach dem Modell der freien Software weiterenwickelt werden. Als z.B. Red Hat 1998 Sicherheitsprobleme in CDE entdeckt, entfernt es den Desktop aus seiner Distribution, da er nicht quelloffen ist und somit diese Fehler nicht einfach behoben werden können.[255]

Hier setzt das *K Desktop Environment* (KDE)[256] an. Das Projekt wurde 1996 von Matthias Ettrich ins Leben gerufen. Die KDE 1.0 wurde im Juli 1998 freigegeben. KDE bietet ein konsistentes *Look-and-Feel* des Desktops und der darauf laufenden Applikationen, in dem sich jede Windows95- oder Mac-Nutzerin sofort zuhause fühlen wird. Eine Programmstartleiste erlaubt den Zugriff auf die Applikationen und auf die aktuell geöffneten Fenster. Drag-and-Drop wird unterstützt wo immer möglich. Der konfigurierbare Fenstermanager *kwm* unterstützt, wie in Unix üblich, mehrere virtuelle Desktops. Für die Konfiguration von Programmen bietet KDE graphische Dialoge. Der Dateimanager *kfm* fungiert gleichzeitig als Web-Browser, über den auch das umfangreiche HTML-basierte Online-Hilfesystem bedient werden und auf Dateien auf entfernten FTP- und WWW-Servern so zugegriffen werden kann, als lägen Sie in einem lokalen Dateisystem. Die Bildschirmtexte sind in über dreißig Sprachen internationalisiert, darunter Mazedonisch, Isländisch und Bretonisch. Alle Nicht-KDE-Applikationen für Unix laufen problemlos auf einem KDE-Desktop, und umgekehrt laufen die meisten KDE-Applikationen auch auf anderen Desktops.

Neben den eigenen Bibliotheken verwendet KDE die C++-Klassenbibliothek Qt der norwegischen Firma Troll Tech AS. Diese Entscheidung traf in der freien Szene auf Kritik, da Qt ein proprietäres Produkt war. Wie meistens, wenn eine nützliche, aber proprietäre Software eine freie Entwicklung verunmöglicht, finden sich Leute, die die betreffende Funktionalität von Grund auf neu schreiben, in diesem Fall das Projekt GNOME (1998). Der Desktop-Krieg zwischen KDE und GNOME[257] führte u.a. dazu, daß Troll Tech seine Lizenzpolitik änderte. Ab Ver. 2.0 liegt die quelloffene Version von Qt unter einer eigenen Lizenz, der Qt Public License (QPL), vor, die es für Open Source-Entwicklung auf Unix-Systemen zur freien Verfügung stellt und für kommerzielle Entwicklungen den Erwerb einer kommerziellen Lizenz vorschreibt. Die QPL hat die meisten Linux-Distributoren dazu veranlaßt, KDE in ihren Paketen aufzunehmen. Nur die in Bezug auf Lizenzpolitik sensibelste Distribution Debian schließt es weiterhin aus.[258] Die KDE-eigenen Bibliotheken und die KDE-Applikationen werden unter der Library GPL veröffentlicht. Das KDE-Manifest bekennt sich ausdrücklich zur kommerziellen Verwendung.[259]

KDE läuft auf einer Vielzahl von Unix-Systemen, darunter Solaris, Irix, HP-UX, AIX, Linux und den BSD-Varianten. Es ist Bestandteil von diversen Linux-Distributionen (darunter SuSE Linux, Caldera OpenLinux und DLD Linux). Die freie C-/C++-Entwicklungsumgebung KDevelop 1.0 wurde im Dezember 1999 offiziell freigegeben. Das

---

[255] s. den Bericht in Linux Weekly News: http://lwn.net/1998/1001/a/cde.html

[256] http://www.kde.org

[257] s. die Slashdot-Diskussion dazu: http://slashdot.org/features/9807150935248.shtml

[258] Carter 6/2000

[259] http://www.kde.org/kde-manifesto.html

für Herbst 2000 angekündigte KDE 2.0 soll u.a. Unterstützung für Java und JavaScript sowie für Unicode anbieten und das neue Paket K-Office enthalten.

KDE ist mit über zwei Millionen Codezeilen eines der umfangreichsten freien Projekte. Mehr als 200 Entwickler aus zahlreichen Ländern arbeiten aktiv daran mit, darunter Deutschland, Norwegen, USA, Kanada, Argentinien, Namibia und Australien. Die gesamte Entwicklung und Verwaltung findet über das Internet mit Werkzeugen wie Mailing-Listen, einem verteilten Versionskontrollsystem (CVS) und diversen frei verfügbaren Programmen zur Konfigurationsverwaltung statt. Das CVS lief anfangs an der Medizinischen Universität zu Lübeck und ist im Februar 2000 an die SourceForge (VA Linnux) in den USA umgezogen. Die verschiedenen Mailing-Listen laufen auf einem zentralen KDE-Server an der Universität Tübingen. Dazu gehören: eine Liste für Anwender, auf denen diese sich gegenseitig selbst helfen, die aber auch von den Entwicklern mitgelesen wird, um gegebenenfalls Hilfestellung leisten zu können; eine Liste für Entwickler, für die Schreibberechtigung auf Anfrage erteilt wird und die für Fragen rund um Entwurf und Entwicklung gedacht ist; eine Liste für die Autoren und Übersetzer der Dokumentation; eine Liste für Lizenzfragen und schließlich eine geschlossene Mailing-Liste, auf der das Kern-Team die allgemeine Richtung von KDE sowie Fragen des Marketings und der Öffentlichkeitsarbeit diskutiert. Aufnahme in das Kern-Team erfolgt aufgrund von fortwährenden Verdiensten um die Weiterentwicklung von KDE.

Laut Kalle Dalheimer, einem der Hauptentwickler, liegt einer der wichtigsten Gründe für den Erfolg von KDE, eine ständig wachsende Zahl engagierter Entwickler anzuziehen, in den Bibliotheken, die es möglich machen, attraktive, mit vielen Features versehene Applikationen zu schreiben, ohne umfangreiche Projektkenntnisse haben zu müssen. Viele Funktionen lassen sich mit nur wenigen Codezeilen integrieren. Damit macht KDE den Einstieg in die Programmierung von Anwendungen mit grafischer Benutzeroberfläche deutlich einfacher. KDE-Entwickler können die erforderlichen Kenntnisse zu einem großen Teil während der Arbeit an ihrem Teilprojekt erwerben, was natürlich viel motivierender ist, als sich erst lange mit den Strukturen vertraut machen zu müssen, bevor man in die eigentliche Entwicklungsarbeit einsteigen kann.[260]

Apache

1989 erfand Tim Berners-Lee am CERN das *World Wide Web* (WWW). Ursprünglich als Werkzeug für die Vernetzung der Informationsflut am europäischen Hochenergiephysikzentrum gedacht, mit öffentlichen Mitteln entwickelt und daher selbstverständlich Open Source, verbreiteten sich das *HyperText Transfer Protocol* (HTTP) und die Seitenbeschreibungssprache *HyperText Markup Language* (HTML) rasch im Internet. Das W3 brachte erstmals eine ähnliche Benutzerfreundlichkeit ins Internet, wie sie Fenster-und-Maus-Oberflächen auf dem PC einführten. Berners-Lee schrieb 1990 auch den ersten Web-Server und Client (Browser und Editor). Wenig später entwickelte ein Team unter Leitung von Rob McCool am National Center for Supercomputing Applications (NCSA) der Universität Illinois einen weiteren Webserver. Da ebenfalls mit Steuergeldern finanziert, war auch diese Software quelloffen und frei für Modifikationen. In den ersten

[260] Dalheimer, Wizards 7/1999

Jahren des maßgeblich durch das Web-Format vorangetriebenen Internet-Booms war der NCSA-Webserver der meistverbreitete.

Als McCool das NCSA verließ, kam die Entwicklung des Servers ins Stocken. Viele Leute fingen an, eigenständig Fehler darin zu beheben und neue Funktionalitäten einzubauen, doch es fehlte eine Sammelstelle, die die neuen Elemente zusammenführte. Zu diesem Zweck gründete sich 1995 die Apache-Group.[261] Der Name "Apache" ist ein Wortspiel auf "A Patchy Server", der durch zahlreiche Patches veränderte Quellcode des NCSA-Servers. Die erste Version des Apache-Webservers wurde noch 1995 herausgegeben. Ein knappes Jahr später hatte der Apache den NCSA-Webserver von der Spitze verdrängt und war zum meistgenutzen Webserver im Internet geworden. 1998 erreichte der Apache einen Marktanteil von 50%. Im Juli 1999 hatte unter den Webserver im Internet Apache einen Anteil von knapp 62%, Microsoft von 22% und Netscape von etwas über 7%.[262]

Der Apache läuft derzeit auf allen gängigen Unix-Derivaten (Linux, BSD, Solaris, AIX, Irix usw.), auf Windows, Siemens BS2000 und Amiga. Er ist sehr stabil, auch für *Mission critical*-Anwendungen geeignet, leistungsfähig und hat einen großen Funktionsumfang. Der Apache benutzt ein modulares System. Der Kern stellt die wichtigsten Funktionen zur Verfügung, die ausreichen, um normale HTML-Seiten auszuliefern. Weitere Funktionen können in Form von Modulen ergänzt werden. Ein Großteil dieser Module ist schon bei der normalen Standarddistribution des Apache dabei. Derzeit gibt es gut hundert verschiedene Module für den Apache, angefangen von CGI und Protokollierungsfunktionen, die anzeigen, wieviele Leute eine Seite betrachtet haben, bis zu URL-Manipulationsroutinen und *Server-side-Includes* wie Java-Servlets. Drei eigenständige Projekte arbeiten unter dem Dach von Apache. Das PHP-Projekt[263] stellt das das PHP-Modul für die Datenbankanbindung ans Web zur Verfügung. Das mod_perl-Projekt[264] integriert einen Interpreter für Perl in den Apache. Ein ähnliches Modul gibt es auch für Python. Das Jakarta-Projekt[265] ist durch eine Kooperation mit Sun neu hinzugekommen und will einen Open Source *Servelet Engine* für den Apache zur Verfügung stellen. Die Entwickler sind zum Teil Mitarbeiter von Sun, zum Teil die Leuten, die bisher das freie JServ-Modul für den Apache entwickelt haben. Dieses Projekt wird, wie alle Projekte, die irgendwann dazu kommen, unter der Apache-Lizenz vertrieben werden. Mittlerweile gibt es einen großen Markt von Drittanbietern von Modulen für den Apache, einige davon quelloffen, andere proprietär.

Die Apache-Lizenz[266] entspricht der *Open Source-Definition* und ähnelt der von BSD. Der Quellcode ist frei verfügbar und kann ohne Lizenzgebühren sowohl für private, wie kommerzielle Zwecke eingesetzt werden. Es gibt nur zwei Einschränkungen. Wenn eine Firma den Apache modifiziert und weitervertreiben will, muß angegeben werden, daß das Produkt auf dem Apache basiert oder Teile davon in das Produkt eingefloßen sind, und dieses Produkt darf dann nicht Apache heißen. Die weiteren Einschränkungen der GPL, daß auch bei abgeleiteter Software der Quellcode mitvertrieben werden muß, gelten bei der Apache-Lizenz nicht.

---

[261] http://www.apache.org

[262] s. Netcraft Survey, http://www.netcraft.com/survey/

[263] http://www.php.net/

[264] http://perl.apache.org/

[265] http://jakarta.apache.org/

[266] http://www.apache.org/LICENSE-1.1.txt

Die Apache-Group (das *Core Team*) besteht derzeit aus 22 aktiven Mitgliedern, überwiegend aus den USA, aber auch aus Kanada, England, Neuseeland, Italien und drei von ihnen aus Deutschland. Lars Eilebrecht ist einer der drei. Das zentrale Kommunikationsmittel des Projekts ist eine einzige öffentliche Mailingliste mit oft mehr als hundert Mails pro Tag. Daneben gibt es Listen für Mitteilungen sowie interne Listen. Entwickelt wird das, wofür sich aktive Interessenten finden. Leute, die bestimmte Module entwickelt haben, fungieren meist als deren *Maintainer*.

Entscheidungen werden getroffen, indem auf der Mailingliste abgestimmt wird. Gibt es Gegenstimmen gegen eine bestimmte Veränderung der Software, "wird typischerweise das Problem, das jemand damit hat, behoben und, wenn der Patch dann für sinnvoll erachtet wird, irgendwann eingebaut. ... Bei Patches, die eine große Änderung darstellen, ist es typischerweise so, daß sich mindestens drei Mitglieder der Apache-Group damit beschäftigt haben müssen, das heißt, es getestet haben und dafür sein müssen, daß der Patch eingebaut wird. "[267]

Die Apache Software Foundation[268] wurde am 1. Juni 1999 gegründet. Es handelt sich um eine *Not-for-Profit Corporation* mit Sitz in Delaware, USA. Das *Board of Directors* der Apache Software Foundation setzt sich ausnahmslos aus Leuten der Apache-Group zusammen. Ziel der Stiftung ist es, Unterstützung für Open Source-Projekte in Form von Hardware, Netzanbindung und Connectivity bis hin zu finanzieller und rechtlicher Unterstützung anzubieten. Ähnlich wie bei XFree86 soll diese Rechtsform Verträge oder sonstige Vereinbarungen mit Firmen ermöglichen. Sie soll auch weitere Entwickler, vor allem Firmen, motivieren, sich an den Projekten zu beteiligen.

In der Apache-Group waren auch schon vor Gründung der Foundation mehrere Vertreter von Firmen beteiligt. Ein bekanntes Beispiel ist IBM, die den Apache in ihrem Produkt *WebSphere* einsetzen. Sie unterstützen die Entwicklung des Apache mit einem eigenen *Apache-Development Team* mit derzeit etwa 13 bis 14 Mitarbeitern. Auch Siemens liefert den Apache mit ihren BS2000 Systemen als Standard-Webserver aus und hat einen Mitarbeiter dafür abgestellt, dafür zu sorgen, daß Portierungsarbeiten vorgenommen werden, und darauf zu achten, daß zukünftige Versionen des Servers auf BS2000 lauffähig sind. Ein weiteres Beispiel ist Appel. Sie haben selber den Apache auf Rhapsody und dann auf Mac OS X portiert. Einer der Mitarbeiter, die dafür verantwortlich waren, wird Mitglied Apache Software Foundation werden.

GIMP

Das Projekt GIMP (GNU Image Manipulation Program)[269] ist mit dem Ziel gestartet, ein Bildbearbeitungsprogramm ähnlich Adobes Photoshop für Linux zu erstellen. Anfang 1995 begannen die beiden Informatikstudenten Peter Mattis und Spencer Kimball, die Basis des GIMP zu schreiben. Ein Jahr später stellten die beiden die Version 0.54 der Öffentlichkeit vor. Bereits in dieser frühen Version fand der GIMP so große Verbreitung, daß Mitte 1996 immer mehr Leute mitentwickeln und ihre eigenen Features einbauen wollten. Auf der GIMP-Mailingliste wurde der Rauschanteil immer höher, so daß die aktiven Entwickler sich

[267] Lars Eilebrecht, Wizards 7/1999

[268] http://www.apache.org/foundation/

[269] http://www.gimp.org

in eine zweite Liste zurückzogen, die zwar immer noch öffentlich war, ihnen aber die Möglichkeit gab, nicht direkt sachdienliche Äußerungen auf die allgemeine Liste zu verweisen.

Als Window-Manager, der die Knöpfe, Fenster und anderen graphischen Bedienelemente zeichnet, wurde zunächst Motif verwendet. Der Quellcode für diesen proprietären Toolkit ist zwar erhältlich, aber zu einem Preis und unter Lizenzbedingungen, die es für ein freies Projekt ausschließen. Deshalb machten die beiden sich daran, ein eigenes Toolkit namens Gtk zu entwickeln. Als Grafik-Toolkit ist Gtk für alle Software unter X-Windows nützlich, weshalb die Motif-ersetzenden Routinen vom GIMP abgetrennt wurden und Gtk als eigenständiges Projekt weiterläuft. So verwendet auch das 1998 entstandene Projekt Gnome das Gtk-Toolkit.

Nach einigen nicht sehr stabilen Versionen erschien Anfang 1997 GIMP 0.99, der erstmals die Motif-Libraries durch Gtk ersetzte. Im selben Jahr wurde ein IRC-Server für die GIMP-Gemeinde eingerichtet. Auch Teilnehmer anderer Projekte treffen sich gern in Chat-Kanälen, aber GIMP ist eines der wenigen, das es als ein eigenständiges Kommunikationsmittel neben den Mailinglisten benutzt.

Wie mehrfach erwähnt scheuen es Entwickler, ihre Arbeit zu dokumentieren. Und gerade das bereits sehr komplexe GIMP bedarf eines Handbuches, um all seine Möglichkeiten auszuschöpfen. Zwei Benutzer, Karin & Olof S. Kylander, widmeten sich dieser Aufgabe und legten Mitte 1998 das zweihundertseitige *GIMP Users Manual* vor, das inzwischen auf 600 Seiten angewachsen ist. Es erscheint auch in gedruckter Form, ist aber weiterhin im Internet frei unter der OpenContent License verfügbar.[270]

Das Programm selbst steht unter der GPL. Die Schnittstelle stehen unter der LGPL, sodaß die Filter und andere Erweiterungen, die Firmen für GIMP oder Photoshop schreiben, in Binärform angeschlossen werden können, ohne daß der Quellcode dafür offengelegt werden muß.

Mitte 1998 erschien GIMP 1.0. Die lange Verzögerung nach Ver. 0.99 lag daran, daß die beiden ursprünglichen Entwickler ihren Universitätsabschluß machten und von der Bildfläche verschwanden, ohne das Projekt an Nachfolger übergeben zu haben. Aus der Konfusion schälten sich erst nach und nach Leute heraus, die fähig und willens waren, neue Versionen zu veröffentlichen. Im Frühjahr 1999 war die (instabile) Version 1.1 erreicht.

Ende 1998 begann Daniel Eggert, den GIMP zu internationalisieren, d.h. er übersetzte alle ursprünglich englischen Menüs ins Deutsche und stellte vor allem den Quellcode so um, daß man weitere Sprachen sehr einfach einbinden kann. Diese mühsame und verdienstvolle Arbeit tat er auf der Mailingliste kund, wo die Nachricht im Rauschen unterging. Enttäuscht über die Nicht-Reaktion beschloß er, sein eigenes GIMP-Projekt aufzumachen und Mitstreiter aufzufordern, mit ihm zusammenzuarbeiten. Die Aufspaltung bedrohte das Projekt als ganzes und hätte die Anwender verwirrt, welche Variante denn nun die bessere sei. Nach kurzer Zeit konnte der Split abgewehrt werden und Eggerts Änderungen wurden in das Hauptprojekt integriert.

Anders als die meisten Projekte verfügt GIMP über keine formelle Organisationsform, sei es eine Stiftung, ein Verein oder auch nur ein *Core-Team*. Wer auf den Mailinglisten hilfreiche Dinge äußert, gehört dazu. Ein formalisiertes Konfliktlösungsverfahren wie beim Apache gibt es nicht. Änderungen werden auch dann

[270] http://manual.gimp.org/

programmiert, wenn jemand nicht damit einverstanden ist -- und setzten sich dann durch oder nicht.[271]

[271] Marc Lehmann, Wizards 7/1999

Lizenzmodelle

Wie aus den vorangegangenen Projektbeschreibungen zu ersehen war, sind für freie Software die Lizenzen, also die vertraglichen Rechte und Pflichten, unter denen die Urheberrechtsinhaber ihre Werke veröffentlichen, von besonderer Bedeutung. Daher geht dieser Abschnitt näher auf die Dimensionen, die diese Lizenzen regeln, und einige wichtige Beispiele ein.

Software ist in Deutschland nach § 2 Abs. 1 Satz 1 UrhG urheberrechtlich geschützt. Dies betrifft insbesondere das Erstveröffentlichungsrecht (§ 12 UrhG). Die Autorin kann ihr Werk unter frei gewählten Bedingungen veröffentlichen, die in einer Lizenz, also einem Privatvertrag zwischen der Urheberin und dem Verwerter oder direkt dem Endnutzer, festgelegt werden. “Unter einer ‘Lizenz’ versteht man die Einräumung von Nutzungsrechten an Schutzrechten.”[272] Die Modelle der freien Software umgehen das Urheberrecht nicht etwa oder verzichten auf seine Inanspruchnahme, sondern setzen darauf auf, um den offenen, kooperativen Prozeß der Erstellung und Weiterentwicklung abzusichern.

Bei der Installation einer Software, gleich ob frei oder unfrei, erscheint in der Regel auf einem der ersten Bildschirme ein langer Lizenztext. Im Falle von proprietärere Software steht darin, daß der Nutzer nicht etwa das Programm, sondern nur ein eingeschränktes Nutzungsrecht daran erworben hat, daß er maximal eine einzige Sicherungskopie erstellen darf, daß er das Programm nur auf einem einzigen Rechner installieren darf, daß er nicht versuchen wird, es zu *reverse engineeren*, daß der Hersteller keinerlei Haftung und Gewährleistung für sein Produkt übernimmt und dergleichen mehr. Diese in juristischen Fachbegriffen abgefaßten Texte liest normalerweise niemand, tatsächlich schließt man jedoch durch das Anklicken der Option “Akzeptieren” einen Vertrag.[273] Die gleiche Prozedur findet sich auch bei freier Software, nur daß die Vertragsbedingungen hier anders lauten.

In den sechziger Jahren war, wie bereits ausgeführt, alle Software in einem bestimmten Sinne frei. Ökonomisch war sie noch nicht zu einer Ware geworden, daher galt sie auch juristisch nicht als Werk, das den Schutz des Urheber-, rsp. Copyright-Rechts oder des Patentrechts genießen kann. Als 1969 der Marktführer IBM unter der Drohung einer kartellrechtlichen Zerschlagung begann, sein *Bundling* von Hard- und Software aufzugeben, war der Startschuß für die Entwicklung einer eigenständigen Software-Industrie gefallen.

Um die seit zwei Jahrzehnten ausstehende Revision des U.S. Copyright Law vorzubereiten und Richtlinien zu seiner Anwendung auszuarbeiten, berief der amerikanische Congress 1974 die CONTU (*Commission on New Technological Uses of Copyrighted Works*) ein. Wie der Name schon sagt, ging es vor allem darum, das Gesetz den Herausforderungen durch neue Technologien, vor allem durch Computer, anzupassen. Die CONTU empfahl, Computer-Programme zukünftig als ‘literarische’ Werke unter den Schutz des Copyright zu stellen.[274] Die grundlegende Copyright-Reform von 1976 folgte dieser Empfehlung, stellte

---

[272] “Die Bezeichnung eines Softwareüberlassungsvertrages als ‘Lizenzvertrag’ sagt jedoch nichts über das anwendbare Recht aus. Erst aus dem Vertragsinhalt ergibt sich, ob Kauf-, Werkvertrags- oder sonstiges Recht anwendbar ist.” Siepmann 1999: Abs. 47. Auch Software-*Patente* werden lizenziert. Darauf wird an anderer Stelle eingegangen.

[273] ob diese Massenmarktlizenzen oder *Shrink-Wrap-Agreements* gültig sind, ist umstritten. Im letzten Abschnitt dieses Kapitels wird darauf eingegangen.

[274] Ein weiteres Motiv war die Anpassung des Copyright Act an internationale Normen, die den Beitritt der USA zur "Berner Übereinkunft zum Schutze von Werken der Literatur und Kunst" ermöglichte.

jedoch eine spezifischere Computer-Copyright-Regulierung aus, bis die CONTU ihren Abschlußbericht vorlegen würde, was 1978 geschah. In der folgenden Revision von 1980 übernahm die U.S.-Legislative die CONTU-Vorlage wörtlich (mit einer folgenreichen Ausnahme[275]). Dem § 101 USCA wurde die Definition von 'Computer-Programm' hinzugefügt ("a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.") Der neue § 117 USCA regelt die Ausnahmen von den ausschließlichen Rechten der Copyright-Inhaber von Computer-Programmen.[276]

Damit hatten ab 1976 Autoren und Firmen die Möglichkeit, ein Copyright auf ihre Software anzumelden. Dazu bringen sie einen Copyright-Vermerk (©, Jahr der Erstveröffentlichung, Name des Copyright-Besitzers) darauf an, müssen zwei Kopien bei der *Library of Congress* hinterlegen und können ihr Copyright beim *Copyright Office* gebührenpflichtig registrieren lassen. Eine Registrierung ist zwar keine Bedingung für den Copyright-Schutz, gilt jedoch als Nachweis im Streitfall und ist Vorraussetzung für Schadensersatzansprüche bei Copyright-Verletzungen.

Im selben Jahr veröffentlichte Bill Gates den bereits erwähnten *Open Letter to Fellow Hobbyists*, in dem er -- noch ohne Verweis auf das Copyright -- ein ökonomisch-moralisches Recht auf die Verwertung der eigenen Software einklagte: "As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid? Is this fair? ... One thing you do is prevent good software from being written. Who can afford to do professional work for nothing?"[277] Mit dem Copyright und ab 1981 auch dem Patenschutz für Software wurden die Auseinandersetzungen aus der öffentlichen Meinungarena in die Gerichtssäle getragen. In einem notorischen Fall wurde Microsoft zum Gegenstand eines langjährigen Urheberrechtsstreits. Apple hatte 1983 die am Xerox PARC entwickelte grafische Benutzeroberfläche mit Fenstern, Menüs und Maussteuerung als Erster auf den Massenmarkt gebracht. Das Windows, mit dem Microsoft 1985 nachzog, glich auffällig dem Apple-Desktop. Apple verklagte Microsoft, das *Look & Feel* des Macintosh plagiiert und damit gegen sein Copyright verstoßen zu haben. Das Verfahren endete erst 1995 mit einer

---

[275] Wo die CONTU dem 'rechtmäßigen Besitzer' (*rightful possessor*) einer Kopie eines Computer-Programms das Recht zugestehen wollte, unter bestimmten Umständen Kopien anzufertigen, wählte der Congress den Begriff 'Eigentümer' (*owner*). Was die Gesetzgeber dazu bewog ist rechtsgeschichtlich ungeklärt, doch führte es zu einer der bizarresten Episoden in der jüngsten amerikanischen Copyright-Geschichte. In zwei umstrittenen Entscheidungen des Ninth Circuit Berufungsgerichts zeigten sich die Folgen (MAI Systems Corp v. Peak Computer Inc., 991 F.2d 511 (9th Cir. 1993) und Triad Corp. v. Southeastern Express, 95 C.D.O.S.6889 (9th Cir. 1995)). In beiden Fällen waren Firmen verklagt worden, die im Auftrag der rechtmäßigen Lizenznehmer der Software der Kläger Reparatur- und Wartungsarbeiten an deren Computersystemen vornahmen. Dazu mußten sie diese Rechner einschalten, d.h. Programme in deren Arbeitsspeicher laden. Das Gericht entschied, daß es sich dabei um einen urheberrechtsrelevanten Kopiervorgang in das RAM handele. Das Gericht interpretierte nun den Eigentümer, der nach § 117 Copyright Act eine Kopie anfertigen oder durch Dritte anfertigen lassen darf, sofern es für die Verwendung des Programms und für Archivierungszwecke erforderlich ist, als den Inhaber des Urheberrechts. Dieser hatte die Kopie seines Werkes aber nicht an seinen Kunden verkauft, sondern nur lizenziert. Vgl. Nicholson 10/1995 und Katz/Hart 1996.

[276] Kopien, die für den Gebrauch des Programms erforderlich sind, Sicherheitskopien, Vermietung und Verkauf solcher zusätzlicher Kopien und Kopien im Zusammenhang mit Wartung und Reparatur von Computern. S. http://www.loc.gov/copyright/title17/92chap1.html#117

[277] http://www.eskimo.com/~matth/hobby.html

Niederlage von Apple, das sofort erneut gegen das gerade erschienene Windows 95 klagte. Dieser Streit endete schließlich mit einem Vergleich.

Seit Mitte der siebziger Jahre werden Verbreitungsstücke von Computer-Programmen, anders als im Fall von Büchern, in der Regel nicht mehr verkauft, sondern lizenziert. Der Inhaber des Copyright oder in Europa der urheberrechtlichen Verwertungsrechte kann im Kauffvertrag und den Allgemeinen Geschäftsbedingungen die Konditionen und den Nutzungsumfang festlegen, unter denen er die Software seinen Kunden zur Verfügung stellt.

In Deutschland genossen Computer-Programme bis 1993 keinen wirkungsvollen gesetzlichen Urheberrechtsschutz. Seither sind Computer-Programme explizit dem Schutz für Sprachwerke (§ 2 Abs. 1 UrhG) unterstellt. Mit der Computer-Rechtsnovelle von 1993 ist eine Richtlinie des Europäischen Rates umgesetzt worden und die §§ 69a bis 69g "Besondere Bestimmungen für Computerprogramme" in das Urheberrechtsgesetz eingefügt worden. Zum Schutz von Datenbankherstellern folgten 1998 die §§ 87a bis 87e UrhG.[278]

Auf die aktuellen Entwicklungen im Bereich der kommerziellen Lizenzen wird am Ende dieses Abschnitts eingegangen. Hier sollen die Lizenzen der freien Software behandelt werden. Dazu ist es zunächst erforderlich, *Free Software* und *Open Source Software* von den benachbarten Konzepten *Freeware*, *Shareware* und *Public Domain Software* abzugrenzen, die häufig zu Verwirrung führen.[279] Wie die englischen Begriffe andeuten, stammen alle genannten Konzepte aus dem US-amerikansischen Kultur- und Rechtsraum. Auf die Übertragbarkeit von *free software* und *open source* auf die deutschen Rechtsverhältnisse wird unten eingegangen.

Shareware ist copyright-geschützte Software, die von ihrem Autor (in der Regel ohne Quellcode und ohne Veränderungsmöglichkeit und -Erlaubnis) kostenlos, aber mit der verbindlichen Bitte veröffentlicht wird, ihm bei regelmäßiger Nutzung des Programms einen bestimmten oder beliebig höheren Geldbetrag zukommen zu lassen. Da sie dem Nutzer erlaubt, das Programm weiterzugeben und auszuprobieren, bevor er es in täglichen Gebrauch nimmte, überschneidet sich die Kategorie mit der *Demoware*, aka *Crippleware*, die einen gegenüber der Vollversion eingeschränkten Leistungsumfang bietet.

Freeware ist copyright-geschützte Software, die von ihrem Autor (in der Regel ohne Quellcode und ohne Veränderungsmöglichkeit und -Erlaubnis) kostenlos, frei weitergebbar und oft ohne Lizenzbedingungen veröffentlicht wird.[280]

---

[278] Siepmann 1999: Abs. 56

[279] Für einen Überblick s. FSF, Categories of Free and Non-Free Software, http://www.gnu.org/philosophy/categories.html. Weiteres Material dazu, aber keinen Ausweg aus der Verwirrung bietet: Gehring 1996. (Zum Unterschied von 'Freeware' und 'Free Software' heißt es dort z.B.: "Diese Begriffe sind nicht notwendig identisch in ihrer Bedeutung, werden aber oft synonym verwendet. Die Feinheiten in der Unterscheidung sind allerdings fallspezifisch und lassen sich nur schwer nachweisen.") Das gleiche Begriffswirrwarr findet sich in der Einleitung von Metzger/Jäger 1999. ("Der Begriff Public Domain Software wird also für eine Vielzahl von Vertragsgestaltungen benutzt...", "... bei der Freeware sei auch eine Veränderungsbefugnis des Nutzers eingeräumt.") Da sich Gehring und Metzger/Jäger z.T. auf dieselbe deutschsprachige juristische Fachliteratur beziehen, ist anzunehmen, daß es sich um eine Art Stille-Post-Phänomen handelt.

[280] Freeware und Shareware finden sich in Sammlungen auf Websites und in kommerziell vertriebenen oder Zeitschriften beigelegten CDs, s. z.B. die CD "freeware, shareware 1", in c't, 12/2000

Public-Domain-Software schließlich ist nicht copyright-geschützt,[281] entweder weil sie gesetzlich nicht schützbar ist[282] oder weil der Autor auf sein Copyright verzichtet,[283] dieses verfallen oder aus formalen Gründen verwirkt worden ist. Das Werk (mit oder ohne Quellcode) wird dadurch gemeinfrei. Jede Nutzung bis hin zur Beanspruchen eines Copyrights durch einen Dritten ist zulässig.

Free software und das jüngere Konzept der *open source software* unterscheiden sich dadurch von den genannten drei Kategorien, daß sie das Copyright/Urheberrecht der Autoren in Anspruch nehmen und zugleich in ihren Lizenzen spezifische Nutzungsfreiheiten festschreiben. Die Details variieren, doch die drei zentralen Punkte, die von den verschiedenen Lizenzen geregelt werden, betreffen die Beifügung des Quellcodes zum Binärcode der Software, das Recht, Kopien anzufertigen und weiterzugeben und das Recht, die urpsprüngliche Software zu modifizieren und die abgeleitete Software zu verbreiten.

BSD-Lizenz

Eine der ersten Quellcode-Lizenzen war diejenige, unter der AT&T sein Unix an Universitäten verkaufte. Sie ging auf eine Lizenz zurück, unter der die Bell-Labs Telefonherstellern Software zur Verfügung stellten. Diese Firmen durften AT&Ts Software benutzen und auch modifizieren, aber sie nicht weiterverbreiten. Das gleiche Lizenzmodell findet man im frühen Unix.[284]

Als die Berkeley Universität begann, Unix-Versionen zu verbreiten, die eigenen Code zusammen mit dem von AT&T enthielten, erarbeiteten die Anwälte von AT&T und der Universität 1979 zu diesem Zweck eine Lizenz.[285] Sie beginnt mit einem Copyright-Vermerk und erlaubt die Verwendung und Weiterverbreitung der Software in Quell- und Binärform, mit oder ohne Veränderungen, solange der Copyright-Vermerk und der Lizenztext mitverbreitet wird, in allen Werbematerialien der Satz “This product includes software developed by the University of California, Berkeley and its contributors” genannt und der Name der Universität und seiner Kontributoren nur mit schriftlicher Genehmigung verwendet wird, um für abgeleitete Software Werbung zu machen. Schließlich folgt ein auch in proprietärer Software üblicher Passus, in dem alle Garantie- und Haftungsansprüche, die sich aus der Verwendung der Software ergeben könnten, zurückgewiesen werden.[286]

---

[281] das Antonym von *public domain* ist *intellectual property*. Etwas in der *public domain* ist also kein Eigentum von jemandem, sondern gehört allen.

[282] Nach deutschem Recht genießen Gesetze, Verordnungen, amtliche Erlasse und Bekanntmachungen sowie Entscheidungen und amtlich verfaßte Leitsätze zu Entscheidungen keinen urheberrechtlichen Schutz (§ 5 Abs. 1 UrhG). Abs. 2 nennt “andere amtliche Werke, die im amtlichen Interesse zur allgemeinen Kenntnisnahme veröffentlicht worden sind”, wobei hierunter u.U. auch Software vorstellbar ist.

[283] Das angloamerikanische Copyright-Recht erlaubt eine solche pauschale Abtretung der Rechte am geistigen Eigentum, das kontinentaleuropäische *droit d’auteur* nicht. Ein Autor, der sein Programm zu freier Software machen möchte, kann hier nach § 31 Abs. 1, 2 UrhG ein Nutzungsrecht an jedermann einräumen, eine Übertragung des Urheberrechts ist, außer an die Erben, nicht zulässig (§ 29 UrhG).

[284] so Borchers, Wizards 7/1999

[285] Die BSD-Lizenz bezog sich ausschließlich auf den an der Universität entwickelten Code. Für eine vollständige Unix-Distribution mußten die Nutzer darüberhinaus die Unix-Source-Code-Lizenz von AT&T erwerben.

[286] s. die Lizenz von 4.4BSD von 1994: http://www.freebsd.org/copyright/license.html

Diese einfache freie Software-Lizenz hat die Lizenzen einer Reihe anderer Projekte, wie das MIT-X-Window System, XFree86 und Apache inspiriert und wird heute noch im Wesentlichen unverändert von den freien BSD-Versionen verwendet. Sie enthält jedoch eine Unklarheit und ein praktisches Problem. Sie schreibt nicht explizit vor, daß abgeleitete Software ebenfalls im Quellcode verfügbar sein muß. Sie muß zwar unter dieselbe Lizenz gestellt werden, insofern vererbt sich auch die Freiheit, die abgeleitete Software im Quellcode weiterverbreiten zu dürfen, doch was geschieht, wenn eine Firma die Quellen für ihre Erweiterungen gar nicht erst veröffentlicht, läßt der Text offen.

Das praktische Problem ergibt sich aus der *Advertising Clause*. Da eine ganze Reihe anderer Projekte diese Klausel übernahmen und die Berkeley Universität durch ihren Namen ersetzten, muß jede Werbung für eine Kompilation solcher Programme eine entsprechend große Zahl dieser Hinweise enthalten. In einer NetBSD-Version von 1997 zählte Richard Stallman 75 Programme, deren Lizenzen jeweils die Angabe eines solchen Satzes vorschrieben. Stallman überzeugte den Rektor der Berkeley Universität von dem Problem, so daß die BSD-Lizenz ab 1999 diese Klausel nicht mehr enthält.[287]

In der BSD-Lizenz kamen zwei Faktoren zusammen. Zum einen durfte AT&T, als Privatunternehmen, dem der Staat ein Monopol in der Telegrafie und Telefonie zugewiesen hatte, sich aus kartellrechtlichen Gründen nicht in andere Wirtschaftsbereiche ausweiten. Es konnte also bis 1984 Unix und andere Software nicht als eigenständige Ware vermarkten. Zum anderen herrscht in den USA die Auffassung, daß die Ergebnisse öffentlich finanzierter Forschung und Entwicklung der Allgemeinheit gehören. Auch die Berkeley Universität wäre daher nicht in der Lage gewesen, ihre Unix-Entwicklungen kommerziell zu vertreiben. Die BSD-Lizenz bekräftigte insofern nur Rechte, die die Allgemeinheit nach vorherrschender Rechtsauffassung ohnehin hat, fixierte das ab 1976 möglich gewordene Copyright an Berkeley-Software und verhinderte, daß ein Dritter diese Software geringfügig veränderte und unter restriktiven Konditionen weiterverbreitete. Berkeley-Code und alle darauf aufbauende Software sollte immer frei weitergebbar und modifizierbar bleiben.

GNU General Public License

> "To copyleft a program, first we copyright it; then we add distribution terms, which are a legal instrument that gives everyone the rights to use, modify, and redistribute the program's code or any program derived from it but only if the distribution terms are unchanged. Thus, the code and the freedoms become legally inseparable."[288]

Im neu entstehenden Software-Markt der späten Siebziger herrschte im Gegensatz dazu eine nachgerade paranoische Haltung. Jeder Käufer erschien den Firmen als potentieller 'Pirat'. Das ihnen gerade zugestandene Copyright schien ihnen nicht ausreichend, so daß sie selbst die ausführbaren Binärversionen ihrer Software nur herausgaben, wenn ihre Kunden eine Vertraulichkeitsvereinbarung (*Nondisclosure Agreement*, NDA) unterzeichneten.[289] Das war allenfalls praktikabel, solange sie es mit einer überschaubaren Zahl von Industriekunden zu

---

[287] Richard Stallman, The BSD License Problem, http://www.gnu.org/philosophy/bsd.html

[288] FSF, What Is Copyleft?; http://www.gnu.org/copyleft/copyleft.html

[289] so z.B. für die Betriebssysteme der VAX und des 68020

tun hatten. Mit dem PC etablierten sich die sog. Massenmarktlizenzen, auf die am Ende dieses Kapitels eingegangen wird. Den Quellcode hüteten sie ohnehin als Handelsgeheimnis. Veränderungen durch die Nutzer sollten verhindert werden. Eine kooperierende *Community* wurde verboten. Die Regel, die die Eigentümer proprietärer Software etablierten, lautete: “If you share with your neighbor, you are a pirate. If you want any changes, beg us to make them.”[290]

Die Hacker der ersten und zweiten Generation reagierten auf diese Entwicklung, indem sie ihre Software in die *Public Domain* stellten oder indem sie zwar ein Copyright dafür anmeldeten, sie aber als *Freeware* deklarierten, meist ohne sich weitere Gedanken über die Details einer Lizenz zu machen, wenn sie nicht gar ins Lager der proprietären Software wechselten.

Eine, wenn man so will, fundamentalistische Reaktion war die von Richard Stallman. 1984 startete er das GNU-Projekt. Er schrieb viel Software, doch seine wohl folgenreichste Erfindung ist das Copyleft.[291] Als Betriebssystementwickler dachte er nicht nur über die technischen Grundlagen von Software nach, sondern näherte sich auch ihrem rechtlichen Status auf tiefgreifendere Weise als die Verfasser anderer freier Lizenzen. In Zusammenarbeit mit juristischen Beratern der FSF, wie dem Columbia-Professor für Recht und Rechtsgeschichte Eben Moglen,[292] entstand daraus die *GNU General Public License* (GPL).[293]

Die Präambel der GPL beginnt mit der Feststellung: “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.” Um sicherzustellen, daß die Software auch in Zukunft frei bleibt, unterliegen die Freiheiten Bedingungen (“restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights”), und das Hinzufügen weiterer Restriktionen wird untersagt. Die Freiheiten und ihre Bedingungen umfaßen im einzelnen:

1. Die Freiheit, das Programm auszuführen (Ziff. 0).[294]
2. die Freiheit, den Quellcode des Programms wörtlich zu kopieren und zu verbreiten, sofern der Copyright-Vermerk und die Lizenz mit kopiert und verbreitet wird. Die Erhebung einer Gebühr für die physikalische Übertragung einer Kopie und für andere Dienstleistungen, wie eine Gewährleistung, wird ausdrücklich zugestanden (Ziff. 1).
3. die Freiheit, das Programm zu verändern und diese veränderte Version zu kopieren und zu verbreiten, sofern das abgeleitete Werk Angaben über die Änderung enhält

---

[290] Stallman 1999: 54

[291] die Anregung zu dem Namen geht auf einen Spruch von Don Hopkins aus dem Jahr 1984 oder 85 zurück: “Copyleft -- all rights reversed.” (Stallman 1999: 59)

[292] z.B. Moglen 1999

[293] hier zitiert aus der derzeit gültigen Version 2, June 1991, Coypright 1989, 1991, Free Software Foundation, Inc., http://www.gnu.org/copyleft/gpl.html. Deutsche Übersetzung s. http://www.suse.de/doku/gpl/gpl-ger.html und http://agnes.dida.physik.uni-essen.de/~gnu-pascal/gpl-ger.html, verbindlich ist jedoch nur das englische Original.

[294] Merkwürdigerweise heißt es im Satz davor, daß andere Aktivitäten außer Kopieren, Verbreiten und Verändern außerhalb des Geltungsbereichs der Lizenz liegen.

und gebührenfrei und unter denselben Lizenzbedingungen[295] veröffentlicht wird, wie das ursprüngliche Programm. Von der Bedingung ausgenommen sind Teile des veränderten Programms, die unabhängige Werke darstellen und separat verbreitet werden (Ziff. 2).

4. die Freiheit, das Programm oder abgeleitete Versionen in Objektcode oder ausführbarer Form zu kopieren und zu verbreiten, sofern der dazugehörige maschinenlesbare Quellcode oder ein schriftliches, mindestens drei Jahre gültiges Angebot, diesen Quellcode auf Anfrage bereitzustellen, beigefügt ist (Ziff. 3).

Die weiteren Sektionen der GPL betreffen den Verfall der Lizenzrechte durch Verstöße (Ziff. 4.), das Verbot, den Empfängern der Software irgendwelche weitergehende Restriktionen aufzuerlegen (Ziff. 6.), Konflikte mit anderen (z.B. Patent-) Ansprüchen, die dazu führen können, daß das Programm nicht weiterverbreitet werden darf (Ziff. 7), mögliche landesspezifische Restriktionen, die dazu führen können, daß diese Länder von der Verbreitung des Programms ausgeschlossen werden (Ziff. 8), mögliche Änderungen der GPL durch die FSF (Ziff. 9) und schließlich den üblichen Gewährleistungs- und Haftungsauschluß, in dem Maße es das anwendbare Recht zuläßt[296] (Ziff. 11 und 12).

Sektion 5 erläutert, wie dieser Lizenzvertrag zwischen Copyright-Inhaber und Nutzer zustande kommt. Darin heißt es, durch die Veränderung oder Verbreitung des Programms oder abgeleiteter Werke zeige der Nutzer seine Einwilligung in die Lizenzbedingungen an.

Sektion 10 eröffnet den Copyright-Inhabern die Möglichkeit, zu entscheiden, ob sie erlauben wollen, daß Teile ihres Programms in andere freie Programme integriert werden, die nicht unter der GPL stehen. Grundsätzlich ist eine enge Kopplung von GPL-Software nur mit anderer Software zugestanden, die ebenfalls unter der GPL steht.[297] Diese Sektion erlaubt Ausnahme nach Einzelfallentscheidung durch die Copyright-Inhaber. Die Kopplung von GPL-Software mit Software-Bibliotheken, die nicht im Quelltext veröffentlicht werden, regelt eine eigenständige Lizenz, die LGPL (s.u.).

Stallmans Ziel mit dieser juristischen Konstruktion war es nicht, Software einfach zu verschenken (wie es bei *Freeware* oder *Public Domain Software* geschieht; er betont, daß GNU nicht in der *public domain* sei, was es erlauben würde, alles damit zu machen, auch die weitere Verbreitung zu beschränken), sondern systematisch einen Bestand an nützlicher Software aufzubauen, der für alle Zeiten (genauer: für die Laufzeit der Schutzfrist des am längsten lebenden Urhebers) frei bleiben wird.

Das klingt nach einem ideologischen, utopistischen Programm. Tatsächlich sind die Beweggründe dahinter ganz pragmatisch: Techniker wollen Dinge erledigen und Probleme lösen, ohne daran durch legalistische Mauern gehindert zu werden. Sie hassen Redundanz. Wenn jemand anderes ein Problem gelöst hat, wollen sie nicht die gleiche Arbeit noch einmal machen müssen. Sie wollen ihr Wissen mit anderen teilen und von anderen lernen, und nicht durch NDAs, ausschließende Lizenzen oder Patente daran gehindert werden.

---

[295] der Text der GPL selbst steht nicht unter der GPL, sein Copyright-Vermerk erlaubt explizit, ihn zu kopieren und zu verbreiten, nicht aber, ihn zu verändern.

[296] in einigen Staaten der USA, in Deutschland und anderen Ländern, in denen ein pauschaler Ausschluß unzuläßig ist, werden diese Abschnitte automatisch unwirksam (dazu s.u.).

[297] unberührt davon bleiben unfreie Programme, die zusammen mit GPL-Software in Sammelwerken (*aggregation*) auf einem Speicher- oder Verbreitungsmedium erscheinen (Ziff. 2).

In juristischen Begriffen gesprochen gewährt die GPL auf des Basis des Ausschließlichkeitsrecht des Urhebers ein bedingtes einfaches Nutzungsrecht an jedermann. Die Bedingungen werden in der GPL formuliert. Verstößt ein Lizenznehmer gegen sie, verfallen die Nutzungsrechte automatisch und die Verbotsrechte von Copyright- rsp. Urheberrecht treten wieder in Wirkung.

Die wichtigste Bedingung besteht darin, daß die von einer unter der GPL stehenden Software abgeleiteten Programme ebenfalls unter der GPL stehen müssen. Ziel dieser häufig als 'infektiös' oder 'viral' bezeichneten Klausel ist es, eine Privatisierung von kollektiv erzeugtem Wissen zu verhindern und den Gesamtbestand an freier Software beständig zu erweitern. Sie hat verschiedene Aspekte. Verbesserten Versionen von GPL-Software dürfen nicht anders denn als *Free Software* verbreitet werden. Die MIT-X-Window-Lizenz, der dieser Vererbungsmechanismus fehlt, führte dazu, daß Firmen die Software auf neue Hardware portiert und proprietär geschlossen, d.h. dem offenen kooperativen Entwicklungsprozeß entzogen haben. Wenn Programmierer GPL-Software um eigene oder Werke Dritter, die unter einer unfreien Lizenz stehen, erweitern, kann das Gesamtwerk nur die Freiheiten der restriktivsten Lizenz gewähren. Da die freie Software-Bewegung nicht bereit ist, mögliche technische Vorteile durch den Verzicht auf Freiheiten zu erkaufen, untersagt sie dies. Programmierern, die bei Firmen oder Universitäten angestellt sind und ihre Verbesserungen an die *Community* zurückgeben möchten, sagt ihr Arbeitgeber nicht selten, daß er ihre Werke zu einem proprietären Produkt machen will. Erfährt er jedoch, daß die Lizenz das nicht zuläßt, wird er die verbesserte Software gewöhnlich unter der GPL freigeben, statt sie wegzuwerfen.[298]

Da der GPL häufig eine antikommerzielle Ausrichtung unterstellt wird, ist die meistgehörte Erläuterung, daß Freiheit im Sinne von Redefreiheit, nicht von Freibier gemeint ist. In der Präambel heißt es "When we speak of free software, we are referring to freedom, not price. ... the freedom to distribute copies of free software (and charge for this service if you whish)..." In Ziff. 1 S. 2 wird die Gebührenerhebung für Dienstleistungen im Zusammenhang mit freier Software ausdrücklich zugestanden: „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." Während es von abgeleiteten Werken explizit heißt, daß ihr Autor sie "at no charge" an jedermann lizenzieren muß (Ziff. 2 b), enthält die GPL keine derartige Aussage über die Verbreitung unveränderter Kopien. Daß über die genannten Dienstleistungen hinaus für die Lizenzierung der Nutzungsrechte allein keine Gebühr erhoben werden darf, wird nirgends expliziert, kann jedoch wohl konkludent angenommen werden.

Nicht nur die Free Software Foundation selbst, sondern viele andere Programmierer sehen in der GPL den besten Mechanismus, um die freie kooperative Weiterentwicklung ihrer Werke zu sichern. So entsteht seit 1984 eine freie Parallelwelt zur proprietären Software, die es heute erlaubt, alles mit einem Computer zu tun, was man möchte, ohne sich den Zumutungen der Unfreiheit aussetzen zu müssen. In der Gesamtschau ist es keine Übertreibung, die GPL als den größten Hack der Wissensordnung zu bezeichnen, seit britische Verlegergilden das Urheberrecht erfanden.

[298] Stallman 1999: 60

GPL und deutsches Recht

Die GPL ist, wie die meisten der hier behandelten Lizenzen, im US-amerikanischen Rechtsraum formuliert worden. Dort ist sie bislang ebenfalls noch nicht gerichtlich überprüft worden, hat aber doch in zahlreichen Fällen zu außergerichtlichen Einigungen mit Software-Unternehmen geführt.[299] Zwar betont Stallman, daß sie den in den Beitrittsländern verbindlichen Auflagen des Berner Urheberrechtsübereinkommens genüge, dennoch ist zu überprüfen, ob sie auch unter den deutschen Rechtsbedingungen einen gültigen Lizenzvertrag darstellt. Hier wird auf die GPL näher eingegangen, da sie vergleichsweise komplexe Regelungen enthält und in der deutschen Rechtsliteratur einige Würdigung erfahren hat. Für die anderen Lizenzen mit sinngemäßen Regelungen gilt gleichermaßen das hier über die GPL Gesagte.

Die Autorin hat auch nach dem deutschen Urherbergesetz (UrhG) ein Ausschließlichkeitsrecht an ihrem Werk, demzufolge sie über seine Veröffentlichung und Verwendung entscheiden kann. Die Kernpunkte der GPL lassen sich ohne weiteres auf das deutsche Urheberrecht abbilden.[300] Die Autorin kann verfügen, daß ihre Software von anderen auf einzelne oder alle Nutzungsarten genutzt werden darf (§ 31, Abs. 1, 2 UrhG, Einräumung von Nutzungsrechten). Dies kann das Vervielfältigungsrecht (§ 16 UrhG), das Verbreitungsrecht (§ 17 UrhG) und das Recht auf Bearbeitungen und Umgestaltungen (§ 23 UrhG) umfassen.

Die Frage, ob durch den impliziten Mechanismus der Ziff. 5 GPL ("By modifying or distributing the Program (..), you indicate your acceptance of this License.") überhaupt ein Vertrag zwischen Urheber und Software-Nutzer zustande kommt, wird von Metzger/Jaeger bejaht. Es handle sich um einen Lizenzvertrag, "den der Nutzer durch die Verwertungshandlungen konkludent annimmt, wobei es eines Zugangs dieser Annahmeerklärung an den Urheber gem. § 151 S.1 BGB nicht bedarf."[301] Siepmann dagegen hält diese automatische Einverständniserklärung für eine Fiktion. "Die Schutzrechte des Urhebers hängen nicht davon ab, ob die Lizenz anerkannt wird."[302] Allerdings behaupt die GPL auch gar nicht, daß die Schutzrechte erst durch das Einverständnis entstehen auf diese wird durch den Copyright-Vermerk am Anfang unmißverständlich hingewiesen. Die Frage scheint vielmehr, ob die Gewährung der Nutzungsrechte und vor allem die daran geknüpften Bedingungen, auf diese Weise vertraglich bindend werden. Auf diesen Punkt wird am Ende des Kapitels bei den Massenmarktlizenzen eingegangen. Auch im folgenden Abschnitt, demzufolge der Empfänger bei jeder Weitergabe des unveränderten und jedes abgeleiteten Programms automatisch eine Lizenz vom Urheber erhält, sieht Siepmann eine Fiktion, die im deutschen Recht keinen Sinn habe.[303]

Die Verknüpfung der Nutzungsrechte an bestimmte Bedingungen (den Quellcode zugänglich zu machen, abgeleitete Werke wiederum unter die GPL zu stellen, Änderungen an Dateien kenntlich zu machen), deren Nichterfüllung den Vertrag automatisch auflöst (Ziff. 4: "Any attempt otherwise [anders, als in der GPL explizit zugestanden] to copy, modify,

---

[299] Powell 6/2000

[300] "Vom Haftungsausschluß abgesehen, ist davon auszugehen, daß die in der GPL gewährten Nutzungsrechte und die Nutzungsbeschränkungen mit dem deutschen Urheberrecht vereinbar sind, also dingliche Wirkung entfalten." Siepmann 1999: Abs. 105

[301] Metzger/Jaeger 1999: V.1.

[302] Siepmann 1999: Abs. 97

[303] Sipemann 1999: Abs. 98

sublicense or distribute the Program is void, and will automatically terminate your rights under this License"), halten Metzger/Jaeger als ein bedingtes Nutzungsrecht i.S.d. § 158 BGB ebenfalls für rechtsgültig. Diese auflösende Bedingung gelte auch für zukünftige abgeleitete Werke, wie die Möglichkeit einer Vorausverfügung des § 40 UrhG zeige. Siepmann hält diesen Abschnitt für rechtlich größtenteils überflüssig.[304]

Die größten Unterschiede zwischen Copyright und Urheberrecht betreffen die Urheberpersönlichkeitsrechte. Während das Recht des Urhebers auf Namensnennung (§ 13 UrhG) durch die Verpflichtung zur Beibehaltung der Copyright-Vermerke in Ziff. 1 und zur Angabe von Veränderungen in Ziff. 2 a) GPL gewahrt wird, scheint der Integritätsschutz des § 14 UrhG problematischer. Danach kann der Urheber Entstellungen und andere Beeinträchtigungen seines Werkes verbieten. Zwar muß ein Bearbeiter nach Ziff. 2 a) GPL seine Änderungen an den entsprechenden Dateien kenntlich machen, dennoch ist der Fall vorstellbar, daß die Verbreitung einer entstellten Version eines Programms zu einer Rufschädigung des Autors der ursprünglichen Version führt. Dies wiegt umso schwerer, als die Ehre, die sich Programmierer für die Qualität ihres Codes in der *Community* erwerben können, eine nicht unmaßgebliche Motivation darstellen. In diesem Fall könnte nach deutschem Recht der ursprüngliche Autor die Verbreitung der abgeleiteten Software verbieten. Metzger/Jaeger kommen zu dem Schluß, "daß es trotz der weitreichenden Freistellung der Open Source Software durch die GPL für den Bearbeiter bei dem Risiko bleibt, sich in Ausnahmefällen einem Verbot des Urhebers auf der Grundlage des § 14 UrhG gegenüberzusehen."

Mögliche Konflikte ergeben sich aus Ziff. 9 GPL, in der die FSF mögliche überarbeitete oder neue Versionen der GPL ankündigt und die Option eröffnet, ein Programm unter eine bestimmte und "any later version" der GPL zu stellen. Der Lizenznehmer könne dann wählen, ob er den Bedingungen der spezifizierten oder der jeweils aktuellen Version folgt. Wird keine Versionsnummer angegeben, kann er eine aus allen je veröffentlichten Versionen wählen. Von der Unsicherheit abgesehen, in welchen Vertrag man eigentlich genau eingewilligt hat, lassen sich Konflikte mit § 31 Abs. 4 UrhG vorstellen: "Die Einräumung von Nutzungsrechten für noch nicht bekannte Nutzungsarten sowie Verpflichtungen hierzu sind unwirksam." Theoretisch sind neue Nutzungsarten denkbar, die in früheren Versionen der GPL nicht expliziert sind und daher Verbotsrechte von Urhebern wirksam werden lassen. Die in Ziff. 9 GPL zumindest implizierte (wenn auch durch die Wahlfreiheit des Lizenznehmers aufgeweichte) Vorausverfügung ist also nach deutschem Recht nicht zulässig. Das Problem kann vermieden werden, indem Autoren die Versionsnummer der GPL, unter der sie ihr Werk veröffentlichen wollen, angeben.

Problematischer stellt sich der generelle Gewährleistungsauschluß (Ziff. 11) -- "to the extent permitted be applicable law" -- und der Haftungsausschluß (Ziff. 12) dar. Bei einem deutschen Lizenznehmer wird das deutsche Recht wirksam, das unabhängig von der vertraglichen eine gesetzliche Haftung[305] vorschreibt, die nicht durch Erklärungen geändert werden kann. Die schuldrechtlichen Fragen sind im Vertragsrecht (BGB) und im

---

[304] Siepmann 1999: Abs. 96

[305] nach § 823 ff. BGB. "Zu den Verkehrssicherungspflichten gehören die Konstruktions-, die Produktions-, die Instruktions-, die Organisations- und die Produktbeobachtungspflicht. Bei der Erfüllung dieser Pflichten ist der jeweilige 'Stand der Technik' zu beachten. Sind Fehler, Schaden und die Ursächlichkeit des Fehlers für den Schaden nachgewiesen, so trifft den Unternehmer die Beweislast dafür, daß kein Verschulden seinerseits vorliegt." (Siepmann 1999: Abs. 68)

Allgemeinen Geschäftsbedinungs-Gesetz (AGBG) geregelt. Hier kommen Siepmann und Metzger/Jaeger zu dem übereinstimmenden Schluß, daß beide Ausschlüsse unwirksam sind. Sie verstoßen gegen die absoluten Klauselverbote des § 11 AGBG. Salvatorische Klauseln wie "Schadensersatzansprüche sind ausgeschlossen, soweit dies gesetzlich zulässig ist" sind nach § 2 Abs. 1 Nr. 2 AGBG unwirksam.[306] Wenn die Nutzungslizenz unentgeltlich eingeräumt wird (eine Schenkung gem. § 516 BGB ist[307]), "ist die Haftung gem. § 521 BGB auf Vorsatz und grobe Fahrlässigkeit beschränkt, die Gewährleistungspflicht für Sach- und Rechtsmängel gem. §§ 523, 524 BGB auf arglistig verschwiegene Fehler."[308] Wird die Software kommerziell in einer Distribution oder vorinstalliert auf einem Rechner vertrieben, sehen Metzger/Jaeger und Siepmann die Anwendbarkeit des Kaufgewährleistungsrechts gegeben.[309] Ob auch das Produkthaftungsgesetz, das sich ausdrücklich nur auf körperliche Gegenstände bezieht, zur Anwendung kommt, ist umstritten. Fielen auch Computer-Programme darunter, würde hiernach eine Haftung sogar unabhängig vom Verschulden bestehen.[310]

Die Gültigkeit der GPL ist weder in Deutschland noch in den USA oder in irgendeinem anderen Land bisher gerichtlich überprüft worden. Konflikte sind in allen Fällen außergerichtlich beigelegt worden (s.u.). Dennoch ist nach der Interpretation der Rechtslage davon auszugehen, daß sich Entwickler, die ihre eigene Software unter die GPL stellen oder sich an GPL-Projekten beteiligen, ebenso wie Endnutzer freier Software ihrer Freiheiten sicher sein können. Vorsicht ist jedoch bei möglichen Haftungsansprüchen geboten. Hier kommt den Autoren und besonders den Distributoren ein Sorgfaltspflicht zu. Sie sollten sich über die Herkunft der verbreiteten Programme, ihre Qualität und mögliche Fehler kundig machen und bei Bekanntwerden von Mängeln ihre Kunden unterrichten. Bei Vernachlässigung ist davon auszugehen, daß sie für überschriebene Dateien, gelöschte Festplatten oder Schäden durch trojanische Pferde haftbar gemacht werden können.

Library GPL

Die GNU Library General Public License[311] der FSF datiert von Juni 1991. Die Präambel erläutert zunächst die GPL und betont, daß die Leserin ihre Software, Bibliotheken eingeschlossen, unter dieser Lizenz ausschließlich für andere freie Programme nutzbar machen kann. Dann führt sie die LGPL als Spezialfall einer Lizenz für "*certain designated*

---

[306] Siepmann 1999: Abs. 55

[307] s. Siepmann 1999: Abs. 44-46

[308] Metzger/Jaeger 1999: VI. 3.

[309] Siepmann sieht Distributionen als eine 'gemischte Schenkung', bei der die dem Händler von Dritten unentgeldlich überlassenen Pakete auch dem Endkunden schenkungsweise überlassen werden, während der Händler die von ihm selbst hergestelleten Bestandteile (z.B. den Installer) verkauft (Siepmann 1999: Abs. 135-140; für den Installer gilt auch dann Kaufrecht, wenn der Distributor ihn unter die GPL stellt, s. ebd.: Abs. 148). Durch den Kaufvertrag für Standardsoftware erhält der Käufer bei Mängeln ein Recht auf Minderung,Wandlung, ev. Ersatzlieferung und ggf. einen Anspruch auf Schadensersatz. Im Falle eines Werkvertrages über die Erstellung von Individual-Software erhält der Käufer davon abweichende Recht, z.B. das auf Nachbesserung und auf Schadensersatz wegen Nichterfüllung (s. ebd.: Abs. 30-36). Einen besonders drastischen Beispielfall, bei dem der Distributor im vollen Umfang für den Schaden durch ein Trojanisches Pferd haftet, s. ebd.: Abs. 142 f.

[310] Siepmann 1999: Abs. 69-70

[311] http://www.gnu.org/copyleft/lgpl.html

libraries" ein, und betont, daß sie sich erheblich von der gewöhnlichen GPL unterscheide. Die LGPL erlaubt die Verwendung von freien Software-Bibliotheken in proprietären Programmen. Als Grund für eine separate schwächere Lizenz gibt die Präambel an, daß einige Bibliotheken die Unterscheidung zwischen Veränderung von Software und ihrem einfachen Gebrauch verschwimmen lassen. "Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such." M.a.W., Ein Programm, das eine Bibliothek unter der GPL linkt, muß selbst ebenfalls unter der GPL stehen. Die libgcc (die mit dem GNU C Compiler verbundene C-*Runtime*-Bibliothek) stand ursprünglich unter der GPL. Daher mußte jedes Programm, das mit der libgcc gelinkt wird, d.h. im wesentlichen jedes Programm, das mit dem gcc kompiliert wurde, wiederum unter der GPL stehen. Da dies die Verwendung des Compilers für viele Entwickler ausschloß, entwickelte die FSF die weniger restriktive LGPL.

Der dahinterliegende Grund ist ein strategischer. "Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better. [...] This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. [...] The hope is that this will lead to faster development of free libraries."[312]

Sehr viel deutlicher äußert sich Stallman zu dieser strategischen Entscheidung in der Begründung für den Namenswechsel von 'Library GPL' zu 'Lesser GPL' im Februar 1999 (s.u.). Dort schreibt er: "free software developers need to make advantages for each other. Using the ordinary GPL for a library gives free software developers an advantage over proprietary developers: a library that they can use, while proprietary developers cannot use it."[313] Grundsätzlich empfiehlt er daher die Verwendung der GPL, doch sei sie nicht für jede Bibliothek vorteilhaft. Vor allem wenn der proprietären Software die *Features* der freien Bibliothek aus anderen, nicht-GPLten Bibliotheken einfach zur Verfügung stünden, wie das bei der GNU C-Bibliothek der Fall ist, brächte die GPL keinen Vorteil für die freie Software, "so it is better to use the Library GPL for that library."

In anderen Fällen bietet die freie Bibliothek bedeutende einzigartige Funktionalitäten, wie die GNU Readline, dann empfiehlt Stallman die GPL. "The Readline library implements input editing and history for interactive programs, and that's a facility not generally available elsewhere. Releasing it under the GPL and limiting its use to free programs gives our community a real boost. At least one application program is free software today specifically because that was necessary for using Readline." In dem Fall fördert die GPL die weitere Entwicklung freier Software. Universitätsprojekte und jetzt, da auch Unternehmen erwägen, freie Software zu entwickeln, auch kommerzielle Projekte können auf diese Weise beeinflußt werden, ihre Software ebenfalls unter die GPL zu stellen.

> "... we can achieve much more if we stand together. We free software developers should support one another. By releasing libraries that are limited to free software only, we can help each other's free software packages outdo the proprietary

---

[312] Präambel, Library GPL, op.cit.

[313] Stallman 1999a

> alternatives. The whole free software movement will have more popularity, because free software as a whole will stack up better against the competition."[314]

Die Grundintention der LGPL entspricht der der GPL. Sie schreibt alle Freiheiten der GPL fest. Die Bibliothek muß frei kopierbar, verbreitbar und modifizierbar sein, der Quellcode von Kopien und Bearbeitungen verfügbar sein, und sie spricht die Urheber ebenso von Haftungs- und Gewährleistungsansprüchen frei. Der Hauptunterschied zur GPL besteht darin, daß Programme, die die freie Bibliothek unter dieser Lizenz einlinken und damit ein ausführbares Ganzes bilden, nicht selbst diesen Freiheiten unterstehen müssen.

Die LGPL betont die Unterscheidung zwischen einem Werk, das von der Bibliothek abgeleitet ist, und einem, das sie benutzt, ohne sie zu verändern. Übersetzungen in eine andere Sprache werden den Veränderungen zugerechnet (Ziff. 0).

Wie unter der GPL müssen Veränderungen angegeben und gebührenfrei an jedermann unter der LGPL lizenziert werden (Ziff. 2 b; c). In die LGPL scheint ein größere Sensibilität für den Integritätsschutz der kontinentaleuropäischen Urheberpersönlichkeitsrechte eingegangen zu sein. Soe heißt es in der Präambel: "If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations." Hinzu kommen die Auflagen, daß das abgeleitete Werk selbst wieder eine Bibliothek sein muß und es in seiner Ausführung nicht von Daten aus einem möglicherweise proprietären Anwendungsprogramm abhängig sein soll (Ziff. 2 a; d). Damit soll verhindert werden, daß LGPL-Bibliotheken derart verändert werden, daß sie nicht mehr von freien Programmen genutzt werden können, da sie nur mit Bestandteilen der proprietären Software zusammenarbeiten.

Section 3 eröffnet die Möglichkeit, die LGPL für eine modifizierte Kopie der Software in die GPL umzuwandeln. Dies sei nützlich, wenn das abgeleitete Werk keine Bibliothek, sondern ein Programm ist, das Teile der Ausgangs-Software enthält. Diese Wandlung ist irreversibel. Umgekehrt ist es nicht möglich, eine Software unter GPL unter die LGPL zu stellen.

Die eigentliche Besonderheit der LGPL sind "works that use the library". Solche Werke für sich sind eigenständig und werden von der Lizenz nicht erfaßt. In dem Augenblick jedoch, wo sie (beim Booten oder zur Laufzeit) die Bibliothek einlinken, entsteht ein ausführbarer Objektcode (ein *executable*), der Teile der Bibliothek enthält, somit ein abgeleitetes Werk darstellt und daher unter die LGPL fällt. Eine Ausnahme bilden abgeleitete Werke, die nur einzelne Elemente einer *Header*-Datei der Ausgangsbibliothek enthalten (numerische Parameter, Datenstruktur-Layouts, kleine Makros und *Inline*-Funktionen "(ten lines or less in length)"). In dem Fall ist die Verwendung des Objektcodes nicht von der LGPL eingeschränkt (Ziff. 5).

Werke, die in diesem Sinne die LGPL-Bibliothek verwenden, dürfen unter Bedingungen eigener Wahl verbreitet werden, sofern diese Bedingungen zulassen, daß der Kunde sie verändert und die Software *reverse engineeren* darf, um diese Veränderungen zu *debuggen*. Wichtigste Voraussetzung dafür ist, daß der Quellcode der Bibliothek zugänglich ist, damit ein Anwender ihn ändern und neu einlinken kann, um ein verändertes *executable* zu

[314] Stallman 1999a

erzeugen.[315] Und natürlich muß das abgeleitete Werk die Copyright-Hinweise und den Lizenztext der Bibliothek enthalten. (Ziff. 6). Der Quellcode der Bibliothek muß, wie bei der GPL mitgeliefert oder drei Jahre lang bereit gehalten werden, doch stärker noch als in der GPL muß der Verbreiter solcher abgeleiteter Werke hier sogar 'verifzieren', daß der Anwender den Quellcode der Bibliothek und den Quell- und/oder Objektcode des Werkes, das die Bibliothek benutzt, erhalten hat (Ziff. 6 d).

Wenn die LGPL den Lizenzbedingungen andere Bibliotheken widersprechen sollte, führt das dazu, daß die sie nicht zusammen in einem Objektcode verwendet werden dürfen (Ziff. 6). Abgeleitete Bibliotheksfunktionen dürfen zusammen mit solchen unter einer anderen Lizenz zu einer Bibliothek kombiniert werden, sofern sie zusammen mit einer nicht-kombinierten Kopie der LGPL-Bibliothek oder einem Hinweis, wo diese zu erhalten ist, verbreitet wird (Ziff. 7).

Die meisten GNU Bibliotheken vor Februar 1999 und ein Großteil der weiteren Bibliotheken, die bei einem Linux-System eingesetzt werden, stehen unter der LGPL. Stallman betont an verschiedenen Stellen, daß es sich um eine strategische Entscheidung handelt, ob ein Entwickler seine Bibliothek unter die GPL oder unter die Library-GPL stellt. "There is no ethical reason to allow proprietary applications on the GNU system, but strategically it seems that disallowing them would do more to discourage use of the GNU system than to encourage development of free applications."[316] Die GNU C-Bibliothek steht unter der LGPL, die sie es erlaubt, auch proprietäre Programme für GNU/Linux zu kompilieren. Bei anderen Bibliotheken müsse man von Fall zu Fall entscheiden.

Auch vorher legte die FSF es Entwicklern nahe, statt der LGPL für ihre Bibliotheken die weitergehende GPL zu verwenden. Ab Februar 1999 ruft Stallman jetzt dazu auf, mehr Bibliotheken unter der GPL zu veröffentlichen[317] und in anderen Fällen die neue Lesser GPL zu verwenden.

Lesser GPL

Die Library GPL ist im Februar 1999 von der GNU Lesser General Public License[318] ersetzt worden (die verwirrenderweise beide mit LGPL abgekürzt werden). Der Namenswechel erfolgte, da 'Library GPL' eine falsche Vorstellung vermittle.[319] Die neue LGPL enthält eine Reihe Änderungen in den vorangestellten Erläuterungen, die nicht Teil der Lizenz sind. So wird als möglicher Grund, statt der GPL die LGPL zu verwenden genannt, daß der weitestmögliche Einsatz der Bibliothek dazu dienen kann, sie als de-facto-Standard zu etablieren. Umgekehrt wäre bei einer freien Bibliothek, die das tut, was auch eine verbreitete unfreie tut, nichts damit gewonnen, sie auf freie Software zu beschränken. Schließlich könne

---

[315] oder noch einmal in der Formulierung von Sebastian Hetze: "Nach den eigentlichen Forderungen ist es so, daß ich ein prä-gelinktes Objekt-File, ein Archiv sozusagen, mit den proprietären Programmodulen ausliefern muß, und dann dieses Objekt-File praktisch gegen jede neue Version der unter der Library-GPL stehenden Library gelinkt werden darf." (Wizards 7/1999-Diskussion)

[316] Stallman 1999: 63

[317] Stallman 1999a

[318] http://www.gnu.org/copyleft/lesser.html

[319] Stallman 1999a

die Erlaubnis, eine freie Bibliothek, wie die GNU C Library, in unfreier Software zu verwenden, dazu führen, daß mehr sie Menschen befähige, einen größeren Korpus freier Software, wie das GNU Betriebssystem oder GNU/Linux zu benutzen.

Der Lizenztext selbst ist weitgehend wortgleich mit der Library GPL. Der wesentliche Unterschied besteht darin, daß unter der Kategorie "Werke, die die Bibliothek benutzen" jetzt auch dynamisch zur Laufzeit eingebundene *shared libraries* erfaßt werden (die neu eingefügte Ziff. 6.b). Auch *Dynamic Link Libraries* (eine von Microsoft zwar nicht erfundene, aber mit seinen DLLs verbreitete Technologie) müssen somit den Tatbestand der Veränderbarkeite und Ersetzbarkeit erfüllen.[320]

Weitere offene Lizenzen

Der ersten Generation von Lizenzen (BSD, GPL und MIT-X) folgten ab etwa 1997 eine Fülle weiterer Lizenzmodelle. Viele Projekte verfaßten eigene Lizenztexte oder variierten bestehende, meist mit einer individuellen Note, aber oft nicht mit der juristischen Sorgfalt, die in die GPL geflossen ist. In jüngster Zeit kommen die Lizenzen von Unterehmen wie Apple, Sun und IBM hinzu, die freien Lizenzen mehr oder weniger ähneln.[321]

Um sich als freie Software oder Open Source-Software zu qualifizieren, muß die Lizenz die Freiheit gewähren, das Programm gebührenfrei und ohne Einschränkungen auszuführen, es zu kopieren und weiterzuverbreiten, und der Zugang zum Quellcode und das Recht, Veränderungen vorzunehmen, müssen gewährleistet sein. Die Hauptunterschiede betreffen die Art, wie die Lizenzen das Verhältnis von freiem zu proprietärem Code und den Status von abgeleiteten Werken regeln.

In den Lizenzen manifestieren sich die politische Philososophie der Projekte und ihr Verhältnis zur Community, zur Wirtschaft und zur Gesellschaft allgemein. Wird Software zu Packages integriert und in Distributionen aggregiert, so treten auch ihre Lizenzen in Wechselwirkung miteinander. Je mehr Lizenzmodelle in Gebrauch sind, desto schwieriger wird die Wahl einer Lizenz und desto unübersichtlicher ihre Wechselwirkungen.

Daher gibt es verschiedene Bestrebungen, Lizenzen nach bestimmten Kriterien zu bewerten und zu klassifizieren. Die FSF unterscheidet sie in ihrer Liste[322] danach, ob es sich um freie Software und um Copyleft handelt und ob eine Lizenz kompatibel zur GPL ist, d.h. ob sie mit GPL-Software gelinkt werden darf. Auch OpenBSD vergleicht in seinem *Copyright Policy*-Dokument eine Reihe anderer Lizenzen und bewertet sie danach, ob die entsprechende Software in die OpenBSD-Distribution aufgenommen werden kann.[323]

---

[320] Ob dieses Bedingungen auch die Software-Bestandteile in den vergleichweise jungen verteilten Objektsystemen erfassen, ist noch ungewiß. Florian Cramer fragt in der Wizards 7/1999-Diskussion (ohne eine Antwort zu erhalten): "Wie verträgt sich diese Vorstellung vom Linken von Code und Kombinieren von verschiedenen Lizenzen eigentlich noch mit verteilten Objektmodellen? Wenn wir z.B. CORBA-Objekte haben, meinetwegen ein CORBA-Objekt aus GNOME, einer GPL-Anwendung, das z.B. über ein anderes Objekt eine Oracle-Datenbank abfragt -- funktioniert das überhaupt noch? Sind diese Vorstellungen des Code-Links nicht eigentlich veraltet und für moderne Programmansätze nicht mehr zu gebrauchen?"

[321] Für einen guten Einblick in die laufenden Diskussionen von Lizenzfragen s. debian-legal: Licensing issues, auf: http://www.debian.org/Lists-Archives/

[322] FSF, Various Licenses and Comments about Them, http://www.gnu.org/philosophy/license-list.html

[323] OpenBSD Copyright Policy, http://www.openbsd.org/policy.html, unter: "Specific Cases"

Peter Deutsch, Autor von Ghostscript, einer weitverbreiteten Implementation von PostScript, erfaßt in seiner Klassifikation[324] *Free Redistribution Licenses* (FRLs). Im Vordergrund steht für ihn die freie Weitergebbarkeit, nicht die Modifikation, weshalb auch Shareware auftaucht. Letztlich geht es ihm darum, seine eigene Aladdin Ghostcript Free Public License (AGFPL) vis à vis den anderen Lizenzmodellen zu positionieren. Deutsch unterscheidet vier Gruppen von FRLs nach ihren Bedingungen für die Weiterverbreitung, den Umständen, unter denen Zahlungen erforderlich sind, und den Bedingungen für den Zugang zum Quellcode. Modifikationsfreiheit ist für ihn kein eigenes Kriterium. Die vier Kategorien sind: (1) Unbeschränkte Lizenzen, z.B. von X-Window, Tcl/Tk,[325] IJG JPEG,[326] PNG/zlib[327] und zip/unzip.[328] Autoren, die eine solche Lizenz verwenden, möchten nur sicherstellen, daß sie als Autoren genannt bleiben, aber stellen darüber hinaus keine Bedingungen. Der Quellcode ist meist verfügbar, doch die Lizenz schreibt seine Mitverbreitung nicht vor. (2) Shareware, die üblicherweise nicht unter freier Software aufgeführt wird, da der Quellcode in der Regel nicht verfügbar ist und bei regelmäßigem Gebrauch eine Lizenzgebühr verlangt wird, aber Deutschs Kriterium ist eben die freie Weitergebbarkeit. (3) Die GNU-Lizenzen GPL und LGPL. (4) *Not-for-profit FRLs*. Deutschs wichtigstes Beispiel hierfür ist die Aladdin (Ghostcript) Free Public License (AFPL).[329] Sie ist von der GPL abgeleitet,[330] setzt aber drei abweichende Akzente. Sie besagt, daß der Quellcode bei jeder Distribution enthalten sein muß, während die GPL in bestimmten Fällen nur einen Pointer auf ihn vorschreibt. Sie untersagt ausdrücklich eine Gebührenerhebung für die Software. “The goal of the AGFPL is to shut out a certain class of GPL ‘free riders’: companies that package GPL'ed software with commercial applications in a way that maintains the integrity of the former while making it seamlessly callable from the latter, which effectively (functionally) makes the GPL'ed software part of the application while respecting the letter of the GPL.”[331] Die Weiterverbreitung von Ghostscript und die Weiterverwendung des Codes in kommerziellen Produkten ist zugestanden, sie erfordert jedoch eine von der AFPL verschiedene gebührenpflichtige Lizenz. Die kommerzielle Version der Software namens Artifex Ghostscript ist identisch zum freien Aladdin Ghostscript, aber Artifex bietet außerdem Support, Bug-Fixing, und Spin-off-Produkte für OEM-Anbieter.[332] Mit dieser Doppellizenzierung vertritt die AFPL die Haltung “... that those who are willing to share should get the benefits of sharing, while those whose own activities operate by commercial rules should have to follow those rules in order to obtain the benefits of software that is [freely redistributable] for others.”[333]

PGP und Minix[334] sind weitere Beispiele aus dieser Kategorie der *Not-for-profit FRLs*. Die FSF bezeichnet sie als ‘halb-freie’ Software. Zwar gewähre sie Individuen alle

---

[324] Deutsch 1996

[325] http://dev.scriptics.com/software/tcltk/license_terms.html

[326] ftp://ftp.uu.net/graphics/jpeg/README

[327] http://swrinde.nde.swri.edu/pub/png/src/libpng-LICENSE.txt

[328] ftp://ftp.uu.net/pub/archiving/zip/doc/LICENSE

[329] ftp://ftp.cs.wisc.edu/ghost/aladdin/PUBLIC

[330] Ältere Versionen von GNU Ghostscript stehen direkt unter der GPL.

[331] Deutsch 1996

[332] http://www.artifex.com/pages/licensing.html

[333] Deutsch 1996

[334] http://www.cs.vu.nl/pub/minix/LICENSE

Freiheiten, daher sei sie viel besser als proprietäre Software. Aufgrund der Restriktionen könne sie aber nicht in eine freie Betriebssystemsumgebung (das GNU-System) integriert werden. Aufgrund der Wechselwirkung der Lizenzen in einem Gesamtsystem würde ein einziges halb-freies Programm das gesamte System halb-frei machen. "We believe that free software should be for everyone -- including businesses, not just schools and hobbyists. We want to invite business to use the whole GNU system, and therefore we must not include a semi-free program in it."[335]

Die Open Source Definition (OSD)[336] ist eine weitere Richtlinie zur Bewertung von Lizenzen. Sie stammt von Bruce Perens, dem langjährigen Maintainer von Debian GNU/Linux. Debian sah sich angsichts der Nachbarlizenzen herausgefordert, genauer zu definieren, was die Freiheit sei, die das Projekt meint. Diese Positionen formulierte Perens nach einer eMail-Diskussion mit den anderen Debian Entwicklern 1997 im *Debian Social Contract*, einem Bekenntnis, daß Debian 100% freie Software bleiben, daß das Projekt alle Neuerungen an die Community zurückgeben und keine Fehler verstecken wird, und in den *Debian Free Software Guidelines* (DFSG).[337] Aus dem Geist des ersten und dem Text der zweiten entstand die OSD. Raymond spielte bei der Universalisierung eine Rolle, da er Perens in seine Bemühungen "das Konzept der freie Software an Leute zu verkaufen, die Krawatten tragen,"[338] eingespannt hatte. Raymond hielt die DFSG für das richtige Dokument, um Open Source zu definieren. Perens entfernte alle Debian-spezifischen Referenzen, tauschte "free Software" gegen "open-source Software" aus und änderte den Namen der Lizenz. Schließlich registrierte er für SPI, der Trägerfirma von Debian, ein *Certification Mark* (CT) auf den Begriff "Open Source". Ein CT ist eine Form von Trademark, eine Art Gütesigel, das den Produkten von Dritten verliehen werden kann.[339] Nachdem Raymond und Perens mit dem dezidierten Ziel, die Open Source-Kampagne und ihr CT zu verwalten, die *Open Source Initiative* (OSI) gegründet hatten, wurde das Eigentum an dem CT von SPI auf die OSI übertragen. Ein gutes Dutzend Lizenzen hat die OSI gutgeheißen und zertifiziert, so daß sie offiziell den geschützten Titel "Open Source™" tragen dürfen.[340]

Die OSD ist, wie gesagt, keine Lizenz, sondern ein Standard, an dem Lizenzen sich messen lassen.[341] Neben den eingangs genannten Freiheiten und den beiden problematischen Punkten, die im Anschluß behandelt werden, enthält die OSD einige Besonderheiten. Während die meisten Lizenzen die Nutzungen ihrer Software ohne Einschränkung an jedermann freistellen, gibt es einige, die explizite Ausnahmen vorsehen. In der Erläuterung zur OSD ver. 1.0[342] führt Perens das Beispiel einer Lizenz der Regents of the University of California, Berkeley an, die die Verwendung eines Elektronikdesign-Programms durch die

---

[335] http://www.fsf.org/philosophy/categories.html

[336] http://www.opensource.org/osd.html

[337] http://www.debian.org/social_contract.html

[338] Perens 1999: 173

[339] "Since the community needs a reliable way of knowing whether a piece of software really is open-source, OSI is registering a certification mark for this purpose, OSI Certified.... If you want to use the OSI Certified mark on your software, you can do this by distributing the software with an approved license from the list and marking the software appropriately." The OSI Certification Mark and Program, http://www.opensource.org/certification-mark.html

[340] OSI, The Approved Licenses, http://www.opensource.org/licenses/

[341] in der fast wortgleichen Fassung der *Debian Free Software Guidelines* ist sie allerdings die Lizenz von Debian GNU/Linux.

[342] Perens 1999: 179

südafrikanische Polizei untersagt. Obgleich das Anliegen zu Zeiten der Apartheit löblich gewesen sei, ist ihr Sinn heute weggefallen, die Lizenzvorschrift für diese und alle abgeleitete Software bestehe jedoch weiter. Ebenso sei es verständlich, daß Autoren den Einsatz ihrer Software in der Wirtschaft, der Genforschung oder einer Abtreibungsklinik untersagen wollten, doch auch diese Anliegen gehörten nicht in eine Lizenz. Deshalb schreibt die OSD für Open Source-Lizenzen vor, daß sie nicht gegen Personen oder Gruppen (Ziff. 5) und gegen Einsatzgebiete (Ziff. 6) diskriminieren dürfen.

Bei der Weitergabe an Dritte soll die Lizenz wirksam sein, ohne daß Eigentümer (der Copyright-Halter) und Lizenznehmer einen Vertrag unterzeichnen (Ziff. 7). Die Gültigkeit von unterschriftslosen Lizenzverträgen wird derzeit auch für den proprietären Bereich diskutiert (s.u.), insofern ist die Erläuterung zur Ziff. 7 der OSD ver 1.0[343] eher ein frommer Wunsch. In der Erläuterung zur ver. 1.7[344] heißt es, daß damit eine Schließung durch zusätzliche Anforderungen wie ein NDA ausgeschlossen werden soll.

Die OSD Ziff. 8 besagt, daß die gewährten Rechte nicht davon abhängig gemacht werden dürfen, daß das Programm Teil einer bestimmten Distribution ist. Es muß frei bleiben, auch wenn es von dieser Distribution getrennt wird.

Die von der OSI zertifizierten Lizenzen sind (Juli 2000)[345]: die GPL und LGPL, die BSD-Lizenz,[346] die MIT- oder X-Konsortium-Lizenz, die Artistic-Lizenz (für Perl entwickelt),[347] die Mozilla Public License (MPL),[348] die Qt Public License (QPL),[349] die IBM Public License,[350] die MITRE Collaborative Virtual Workspace License (CVW License),[351] die Ricoh Source Code Public License,[352] die Python-Lizenz[353] und die zlib/libpng-Lizenz.

Bevor auf die beiden kritischen Punkte eingangen wird, sei noch die Möglichkeit der Mehrfachlizenzierung erwähnt. Wie beim bereits genannten Ghostscript gibt es eine Reihe Projekte, die dieselbe Software geichzeitig unter einer freien und einer kommerziellen Lizenz anbieten. Eine weiteres Beispiel ist Sendmail,[354] das Eric Allman an der Berkeley Universität entwickelt und 1997 eine Firma[355] gegründet hat, die parallel zur freien Version eine kommerzielle Lizenz mit Supportleistungen anbietet. Die MPL ist die einzige Lizenz, die diese Möglichkeit ausdrücklich erwähnt. Ziff. 13 erlaubt es den ursprünglichen Entwickler (nicht aber den Kontributoren), ihren Code unter die MPL und zugleich eine alternative Lizenz zu stellen, aus denen Nutzer ihre Wahl treffen können. Darin ist die Handschrift von

---

[343] Perens 1999: 179

[344] Perens, Rationale for the Open Source Definition, http://www.opensource.org/osd-rationale.html

[345] http://www.opensource.org/licenses/

[346] die ursprüngliche Fassung (noch mit der *Advertizing Clause*) in der Version für 4.4BSD von 1994: http://www.freebsd.org/copyright/license.html; die generische BSD-Lizenz in der derzeit gültigen Fassung: http://www.opensource.org/licenses/bsd-license.html; in der Fassung von FreeBSD fällt schließlich auch noch der Verweis auf die Trägerorganisation weg: http://www.freebsd.org/copyright/freebsd-license.html

[347] http://language.perl.com/misc/Artistic.html

[348] http://www.mozilla.org/MPL/MPL-1.1.html

[349] http://www.trolltech.com/products/download/freelicense/license.html

[350] http://www.research.ibm.com/jikes/license/license3.htm

[351] http://cvw.mitre.org/cvw/licenses/source/license.html

[352] http://www.risource.org/RPL/RPL-1.0A.shtml

[353] http://www.python.org/doc/Copyright.html

[354] http://www.sendmail.org

[355] http://www.sendmail.com

Perens zu erkennen, der denjenigen, die ihre Software frei belassen und sie zugleich verkaufen möchten, eine beliebige kommerzielle Lizenz plus der GPL als freie Lizenz empfiehlt.[356] Eine eigenartige Konstruktion ist die CVW-Lizenz des MITRE. Sie ist nur eine Art Rahmenlizenz, in der die Warenzeichen von MITRE von der Werbung für abgeleitete Werke ausgeschlossen werden. Darüber hinaus stellt sie dem Empfänger der Software frei, ob er sie unter der GPL oder der MPL nutzen möchte, die beide in der CVW-Lizenz enthalten sind.

Verhältnis von freiem und proprietärem Code

Darf freier Code in unfreien integriert werden? Die GPL untersagt dies, bietet aber mit der LGPL eine Ausnahmemöglichkeit für Bibliotheken. Eine weitere Ausnahme ist die Lizenz von Guile,[357] einer erweiterbaren Interpreter-Bibliothek der FSF für die Programmiersprache Scheme. Guiles Lizenz besteht aus der GPL und einem zusätzlichen Passus[358], der die pauschale Erlaubnis gibt, unfreie Software mit der Guile-Bibliothek zu linken, ohne das resultierende *Executable* unter der GPL stehen muß.

Die BSD-[359] und davon abgeleitete Lizenzen[360] (wie die MIT-X-, Open Group X-[361] und die XFree-Lizenz, die des Apache,[362] die von Zope,[363] von Python,[364] PNG/zlib,[365] Tcl/Tk[366] und die von Amoeba[367]), also diejenigen, die Deutsch 'unbeschränkte Lizenzen' nennt, erlauben, alles mit ihrer Software zu machen -- auch, sie in unfreie Software zu integrieren und Änderungen zu privatisieren --, solange der Hinweis auf den Copyright-Halter erhalten bleibt und nicht mit dessen Namen geworben wird.[368] Beide Lizenzfamilien stammen aus Universitäten, und spiegeln die Auffassung wieder, daß öffentlich finanzierte Werke allen ohne Einschränkung gehören.

Der Grund dafür, daß auch außeruniversitäre Projekte diese Lizenzen verwenden, liegt in der Zusammenarbeit mit Unternehmen, die ihr geistiges Eigentum nicht freizügig teilen möchten. Hohndel vom XFree86-Projekt:

> "Die Geschichte von vielen Projekten, nicht nur von X, auch von Dingen wie Sendmail oder BIND, zeigt, daß diese eher abschreckenden Bestimmungen der GPL

---

[356] Perens 1999: 185

[357] http://www.gnu.org/software/guile/guile.html

[358] die "*special exception*" ist zu finden in: http://www.somelist.com/mail.php3/128/view/394823

[359] OpenBSD verwendet die Original-Berkeley-Lizenz und teilweise eine Modifikation, bei der die Werbevorschrift entfernt wurde (http://www.openbsd.org/policy.html).

[360] eine generische *BSD-style* Lizenz s. http://www.debian.org/misc/bsd.license

[361] http://www.x.org/terms.htm

[362] http://www.apache.org/LICENSE.txt

[363] http://www.zope.com/Resources/ZPL

[364] http://www.python.org/doc/Copyright.html

[365] http://swrinde.nde.swri.edu/pub/png/src/libpng-LICENSE.txt

[366] http://dev.scriptics.com/software/tcltk/license_terms.html

[367] http://www.cs.vu.nl/pub/amoeba/COPYRIGHT

[368] FreeBSD umfaßt eine ganze Reihe Ports mit zusätzlichen Einschränkungen, da sie proprietäre Software enthalten, Exportskontrollen unterliegen (Krypto) oder ihre Autoren eine *for-profit*-Verbreitung untersagen. S. http://www.freebsd.org/copyright/LEGAL

> gar nicht nötig sind. Für uns ist der große Grund, warum wir mit einer GPL überhaupt nichts anfangen können, wiederum die Zusammenarbeit mit den Firmen. Denn viele kommerzielle Unix-Systeme enthalten [...] heute X-Server, die im wesentlichen auf XFree86 basieren. Und viele von den Firmen, mit denen wir zusammenarbeiten und von denen wir Source Code bekommen -- es gibt ja etliche Graphikkartenhersteller, die heute selber die Treiber für ihre Karten schreiben, --, würden niemals zulassen, daß eine so virenartige Lizenz wie die GPL, die sich in den Code reinfrißt und nie wieder daraus weggeht, in ihren Code reinkommt. Von daher folgen wir dieser sehr, sehr freien Lizenz, die uns bis jetzt sehr gut gedient hat, und werden das auch weiterhin tun."[369]

Die OSD besagt, daß eine Lizenz andere Software nicht 'kontaminieren' darf (Ziff. 9), d.h. sie darf nicht vorschreiben (wie im Falle einer Version von Ghostscript), daß alle andere Software, die auf dem selben Medium verbreitet wird, ebenfalls freie Software sein muß. In den Erläuterungen zur OSD ver 1.7 heißt es, daß die GPL diese Auflage erfülle, da sich ihr 'Kontaminierungseffekt' nur auf Software bezieht, die mit der GPL-Software zu einem funktionalen Ganzen gelinkt wird, nicht aber auf solche, die nur mit ihre zusammen verbreitet wird.[370]

Die Perl Artistic License erlaubt es ausdrücklich (Ziff. 5), den Perl-Interpreter mit proprietärem Code zu linken und betrachtet das Ergebnis nicht als abgeleitetes Werk, sondern als Aggregation. Auch die Aggregation in einer kommerziellen Distribution ist zugestanden, solange das Perl-Paket 'eingebettet' wird, "that is, when no overt attempt is made to make this Package's interface visible to the end user of the commercial distribution." (Zif. 8) Im Umkehrschluß kann man annehmen, daß andernfalls die Integration in kommerzielle Software verboten ist.

Status abgeleiteter Werke

Dürfen Veränderungen privatisiert werden? Die GPL schließt dies entschieden aus. Der Oberbegriffe "free Software" (der z.B. *Public Domain*-Software beinhaltet) bedeutet, daß sie verwendet, kopiert, mit Quellcode weiterverbreitet und verändert werden darf, schließt aber nicht aus, das Kopien und Modifikationen ohne diese Freiheiten (z.B. ausschließlich als *Executable*) verbreitet werden. Das engere, wenn auch nicht auf die GPL beschränkte "Copyleft" bedeutet darüberhinaus, daß modifizierte Versionen von GPL-Software gleichfalls unter der GPL stehen müssen und keine weitere Nutzungseinschränkungen hinzugefügt werden dürfen. Einmal frei, immer frei.

Von der GPL abgeleitete Lizenzen (wie die Ghostscript AFPL und die Arphic Public License[371] schreiben gleichfalls vor, daß Modifikationen kostenfrei und zu den selben Bedingungen veröffentlicht werden müssen, wie die Ausgangs-Software.

---

[369] Dirk Hohndel, Wizards 7/1999

[370] Perens, Rationale for the Open Source Definition, http://www.opensource.org/osd-rationale.html#clause9

[371] für Fonts der taiwanesischen Arphic Technology Co. ftp://ftp.gnu.org/non-gnu/chinese-fonts-truetype/LICENSE

In der OSD ist diese Lizenzvererbung nur eine Kannvorschrift. Ziff. 3 besagt, daß eine Lizenz zulassen muß, daß abgeleitete Werken unter die selbe Lizenz gestellt werden, in seiner Erläuterung schreibt Perens, daß sie es aber nicht vorschreiben muß. Ziff. 4 OSD enhält zudem ein Schlupfloch für die Veränderbarkeit. Unter der Überschrift "Integrität des Quellcodes des Autors" heißt es dort, daß die Verbreitung modifizierter Versionen des Quellcode eingeschränkt werden kann, aber nur, wenn es zulässig ist, *Patch*-Dateien zusammen mit dem unveränderten Quellcode zu verbreiten, die zur *Build*-Zeit die modifizierte Version erzeugen. Diese Integration von *Patches* automatisieren Werkzeuge, weshalb der zusätzliche Aufwand vertretbar sei. Damit soll eine Trennung zulässig werden, die die Integrität des 'Originals' wahrt und trotzdem Modifikation möglich macht. Die Verbreitung von Objektcode-Versionen darf nicht eingeschränkt werden, die Lizenz kann aber vorschreiben, daß abgleitete Werke einen anderen Namen tragen müssen. Als Grund für diesen Passus nennt Perens, daß er die Qt Public License (QPL) von Troll Tech in die Open Source-Definition aufnehmen wollte. Diese erlaubt Modifikationen ausschließlich in der Form von Patches. Die QPL ist zwar eine freie Lizenz, aber inkompatibel zur GPL. Daher darf Qt nicht mit GPL'ter Software gelinkt werden, doch erteilt die FSF hierfür eine Sondergenehmigung. Der Copyright-Halter eines Programmes, das Qt verwendet, kann sein Programm unter die GPL stellen, indem er folgenden Hinweis hinzufügt: "As a special exception, you have permission to link this program with the Qt library and distribute executables, as long as you follow the requirements of the GNU GPL in regard to all of the software in the executable aside from Qt."[372]

Die unbeschränkten Lizenzen (wie BSD, Apache oder X) erlauben, daß Veränderungen privatisiert werden. So ist es möglich, den Quellcode einer Software unter X-Lizenz zu verändern und Binaries davon verkaufen, ohne deren Quelltext offen zu legen und die modifizierte Version wieder unter die X-Lizenz zu stellen. Tatsächlich gibt es eine Reihe Workstations und PC-Grafikkarten, für die ausschließlich unfreie Versionen von X-Window verfügbar sind.[373]

Die Perl Artistic License verlangt, daß Änderungen kenntlich gemacht werden und bietet dann vier Optionen für das weitere Vorgehen: a) Die Änderungen müssen in die *Public Domain* gestellt oder auf andere Weise frei verfügbar gemacht werden, oder der Copyright-Halter muß sie in die Standardversion aufnehmen dürfen. b) Die Änderungen werden ausschließlich innerhalb eines Unternehmens oder einer Organisation genutzt. c) Die Nicht-Standard-*Executables* erhalten einen neuen Namen und dürfen nicht ohne die Standardversionen verbreitet weren. d) Andere Verbreitungsvereinbahrungen mit dem Copyright-Halter werden getroffen (Ziff. 3). Sie bietet somit einigen Spielraum für Privatisierungen.[374]

---

[372] FSF, Various Licenses and Comments about Them, http://www.gnu.org/philosophy/license-list.html

[373] zum Debakel um die kurzzeitige Schließung der offiziellen Version von X Window durch die Open Group s.o. unter "XFree86".

[374] Spielraum und Schlupflöcher bietet die AL auch sonst. Stallman bezeichnet sie als "too vague", Perens als "sloppily-worded." Deshalb sei jetzt fast alle Software unter der AL doppelt lizenziert und bietet die Wahl zwischen der AL und der GPL. (Perens 1999: 183)

Die CVW-Lizenz des MITRE ist auch hier ungewöhnlich. Ziff. 5 besagt, daß derjenige, der geänderten Quellcode an MITRE übermittelt, einwilligt, sein Copyright daran an MITRE zu übertragen, das die Änderungen auf seiner Website zugänglich macht.[375]

Beispiele für Lizenzkonflikte

Lizenzfragen betreffen vor allem Entwickler und Vertreiber von Software. Anwender, die die Software nicht selbst modifizieren möchten, haben mit jeder der genannten Lizenz die Gewähr, sie (bei den meisten auch für kommerzielle Zwecke) einsetzen und an Freunde weitergeben zu dürfen. Ein Distributor hat vor allem auf die verschiedenen Lizenzbedingungen für die Aggregierung und den Vertrieb zu achten. Auch eine Entwicklerin muß sehr genau entscheiden, welche Lizenz sie für ihre Programme benutzt, und welche Programme sie in ihren eigenen Werken verwendet, da sie mit dem Code auch dessen Lizenz importiert. Die einzelnen Lizenzen in einem zusammengesetzten Werk treten in Wechselwirkung. Kombinationen aus proprietärem und freiem Code sind, wenn die beteiligten freien Lizenzen es zulassen, als ganze proprietär. Untersagt eine freie Lizenz wie die GPL die Verbindung mit unfreiem Code, muß dieser entweder freigegeben werden oder die Verbindung darf nicht durchgeführt werden. In der Praxis sind die Wechselwirkungen jedoch erheblich komplexer und werde durch den Zuwachs an neuen Lizenzmodellen immer schwieriger zu überschauen.

Die GPL ist vergleichsweise eindeutig. Stallman gibt zwei Beispiele für die postiven Auswirkungen ihres Schließungsverbots:

> “Consider GNU C++. Why do we have a free C++ compiler? Only because the GNU GPL said it had to be free. GNU C++ was developed by an industry consortium, MCC, starting from the GNU C compiler. MCC normally makes its work as proprietary as can be. But they made the C++ front end free software, because the GNU GPL said that was the only way they could release it. The C++ front end included many new files, but since they were meant to be linked with GCC, the GPL did apply to them. The benefit to our community is evident.
>
> Consider GNU Objective C. NeXT initially wanted to make this front end proprietary; they proposed to release it as .o files, and let users link them with the rest of GCC, thinking this might be a way around the GPL's requirements. But our lawyer said that this would not evade the requirements, that it was not allowed. And so they made the Objective C front end free software.
>
> Those examples happened years ago, but the GNU GPL continues to bring us more free software.”[376]

NeXT dachte, es könne die Objective C-Erweiterungen zu einem separaten Modul erklären und ohne Quellcode ausliefern. Nach einer Klage der FSF, der Copyright-Inhaberin des GCC, lenkte NeXT ein, und seither erfreut sich die freie Software-Welt der ‘spendierten’ Software. Auch die Firma Be hat GPL-Quellen unrechtmäßig verwendet, was bald entdeckt wurde und ebenfalls zum Einlenken führte. Gleiches geschah im Falle einer Firma, die

---

[375] http://cvw.mitre.org/cvw/licenses/source/license.html

[376] Stallman 1998

Readline, eine Bibliothek, die *Command-Line Editing* ermöglicht und unter der GPL steht, in einem unfreien Programm verwendete. Auch hier stellte der Entwickler sein eigenes Programm auf Nachfragen ebenfalls unter die GPL. Meistens sind jedoch Lizenzverstöße nicht einfach festzustellen -- naturgemäß, wenn die betreffende Firma den Quellcode ihrer abgeleiteten Software nicht freigiebt.

Doch nicht nur Firmen, sondern auch einige freie Projekte, die mit Firmen zusammenarbeiten, haben ihre Probleme mit der GPL. Patrick M. Hausen von BSD:

> "Wenn Sie ein Stück Software, das unter der GPL ist, in ihre eigene Software einbauen, dann wird die gesamte Arbeit, die Sie gemacht haben, automatisch eine aus der GPL-Software abgeleitete Arbeit, und damit gilt die GPL für alles, was Sie getan haben. ... Und so ist es eben schwierig, GPL-te Open Source-Software in kommerziellen Produkten mit anderen Teilen, sei es mit Treibern, die unter *Non-disclosure-Agreement* sind, sei es mit Librarys, mit einer Oracel-Datenbank-Library oder mit Motif oder was immer zu kombinieren, um ein Gesamtes zu entwickeln. Mir bleibt letzten Endes nichts anderes übrig, als den GPL-ten Teil komplett neu zu schreiben."[377]

Auch Kalle Dalheimer von KDE sieht durch die Einschränkung der Verbreitung von *Binaries* in der GPL die Endnutzerfreundlichkeit von Software behindert.

> "Ich kritisiere nicht die GPL, weil sie mit der (derzeitigen) Qt-Lizenz inkompatibel ist, sondern weil sie so gestaltet ist, daß sie die Weiterentwicklung von Linux *behindert*. In den Anfangstagen war sie sicherlich gut und nützlich, aber jetzt hat sich die GPL überlebt -- zumindest für alles, was im weitesten Sinne eine Komponente ist."[378]

Die 'einschüchternde' Wirkung der GPL taucht auch in der Begründung für Netscapes 'permissivere' Mozilla-Lizenz (s.u.) auf:

> "Netscape is interested in encouraging the use and development of the Communicator source code by other commercial developers. Netscape was concerned that these other companies would hesitate to engage in this development if the code were regulated by a license as strict as the GPL, requiring that all related software also be released as free source. At the very least, every other commercial developer would have to look very closely at the legal ramifications of embarking on free source development. This initial hurdle alone might be enough to keep them from starting at all, so it was decided to remove the hurdle by using a somewhat less restrictive license."[379]

Mike Loukides schlägt in O'Reillys Java-Forum ein schärferen Ton an. Die 'Chamäleon-Lizenz, von der er spricht, ist die *Sun Community Source License* (s.u.), die die lizenzierte Software frei mache, wenn sie im Kontext freier Software erscheint, und kommerziell, wenn sie mit proprietärer Software kombiniert wird.

---

[377] Hausen, Wizards 7/1999 Diskussion

[378] Kalle Dalheimer auf de.alt.comp.kde, 1998/11/29

[379] Netscape Public License FAQ, http://www.mozilla.org/MPL/FAQ.html#4

> “With all due respect to RMS, I think it's clear that the GPL significantly retarded acceptance of Open Source software. The GPL was clearly an important statement, but the ideology was way ahead of what most people wanted to practice. We all know people who could not take advantage of GNU software because they didn't understand exactly what the license meant, what they were giving up, what they could conceivably be sued for in a couple of years, and so on. We all know of companies that wouldn't allow any GNU software on their machines because they didn't want to take the risk that someone would borrow a few lines of source code and compromise the company's intellectual property. Whether you agree with this or not isn't the issue; the fact is, it didn't do any good to have a license that made people scared to use the software. [...] The bottom line is that the GPL is fundamentally coercive, and was intended to be so. Morality aside, that just plain hurt the cause. The right way to popularize free software wasn't to threaten people who chose to use it. [...] This is where the chameleon license offers an important new model. It's much more developer-friendly, and can entice developers to the Open Source model. It's a carrot, not a stick. [...] Although you can go the commercial route if you want, Sun has given you an implicit incentive to go the Open Source route--or at least to consider it. [...] It's an opportunity that opens the community's doors to pragmatic developers: people who'd like to figure out how to make some money from their work, and are put off by licensing zealotry. It gives them a real opportunity to experiment with different licensing options without penalty, and gently pushes them in the Open Source direction.”[380]

Stallman antwortet auf diese ‘Angst vor Freiheit’:

> “The main argument for the term ``open source software" is that ``free software" makes some people uneasy. That's true: talking about freedom, about ethical issues, about responsibilities as well as convenience, is asking people to think about things they might rather ignore. This can trigger discomfort, and some people may reject the idea for that. It does not follow that society would be better off if we stop talking about these things. [...] Years ago, free software developers noticed this discomfort reaction, and some started exploring an approach for avoiding it. They figured that by keeping quiet about ethics and freedom, and talking only about the immediate practical benefits of certain free software, they might be able to ``sell" the software more effectively to certain users, especially business. The term ``open source" is offered as a way of doing more of this--a way to be ``more acceptable to business." This approach has proved effective, in its own terms. Today many people are switching to free software for purely practical reasons. That is good, as far as it goes, but that isn't all we need to do! [...] Sooner or later these users will be invited to switch back to proprietary software for some practical advantage. Countless companies seek to offer such temptation, and why would users decline? Only if they have learned to value the freedom free software gives them, for its own sake. It is up to us to spread this idea--and in order to do that, we have to talk about freedom. A

[380] Loukides o.J.

> certain amount of the ``keep quiet'' approach to business can be useful for the community, but we must have plenty of freedom talk too.”[381]

In diesem Geist ist die Welt der freien Software immer bestrebt, proprietäre Angebote für wesentliche Systembestandteile durch freie zu ersetzen oder die Rechteinhaber zu überzeugen, ihre Software unter eine freie Lizenz zu stellen. Klassisches Beispiel für letzteres ist die mehrfach angesprochene Qt-Bibliothek der Firma Troll Tech, auf die sich der freie Desktop KDE stützt. Troll Tech hat auf Drängen der Linux-Welt Qt ab Ver. 2.0 unter eine eigene Lizenz, die Q Public License (QPL)[382] gestellt, die seine proprietäre Verwendung ausdrücklich ausschließt. Die QPL läßt Modifikationen nur separat vom Original in Form von *Patches* zu und gewährt Troll Tech ihre Verwendung in anderen Produkten (Ziff. 3). Programme, die mit der Qt-Bibliothek linken, müssen selbst quelloffen sein (Ziff. 6). Die QPL ist eine *Not-for-profit*-Lizenz. Wer Qt kommerziell einsetzen möchte, muß die *Professional Edition* unter einer herkömmlichen kommerziellen Lizenz erwerben. Troll Tech hat die Copyright-Rechte an der Qt Free Edition und das Recht, die QPL zu ändern, an die im April 1998 errichtete *KDE Free Qt Foundation*[383] abgetreten -- ein außergewöhnlicher Schritt eines Unternehmens auf die freie Community zu:

> “Should Trolltech ever discontinue the Qt Free Edition for any reason including, but not limited to, a buyout of Trolltech, a merger or bankruptcy, the latest version of the Qt Free Edition will be released under the BSD license. Furthermore, should Trolltech cease continued development of Qt, as assessed by a majority of the KDE Free Qt Foundation, and not release a new version at least every 12 months, the Foundation has the right to release the Qt Free Edition under the BSD License.”[384]

In den Linux-Distributionen von SuSE oder RedHat ist die Bibliothek enthalten, was zu dem Mißverständnis führen kann, daß man sie in eigenen Entwicklungen einsetzen darf. Tut man dies, hat man jedoch eine Lizenzklage und entsprechende Folgekosten zu gewärtigen.

> “SuSE hat ohnehin Qt aufgeteilt in die Bibliothek selbst und in die Header-Dateien. Und wenn Sie während der Installation eines SuSE-Systems nur die Bibliothek installieren, passiert nichts weiter, und Sie können dann ja auch nicht aufs Glatteis kommen. In dem Moment, wo Sie das Paket mit den Header-Dateien ausfrieren, poppt so eine kleine Box hoch, wo drin steht: 'Beachten Sie bitte die Lizenzbedingung'.”[385]

Weitere Beispielfälle für denkbare urheber-, vertrags-, patent-, markenschutz-, und haftungsrechtliche Konflikte, die in den Beziehungen zwischen Autoren, Distributoren und Endnutzern auftreten können, gibt Siepmann.[386]

---

[381] Stallman 1999b

[382] http://www.trolltech.com/products/download/freelicense/license.html. Die QPL gilt für die Qt Free Edition ab Version 2.0. Ältere Versionen stehen unter der nicht mehr verfügbaren QT Free Edition License.

[383] http://www.de.kde.org/kdeqtfoundation.html

[384] http://www.trolltech.com/company/announce/foundation.html

[385] Dalheimer, Wizards 7/1999 Diskussion

[386] Siepmann 1999: Abs. 121 - 157

Unfreie Lizenzen

Auf aktuelle Entwicklungen bei den rein proprietären Lizenzen kommt der folgende Abschnitt zu sprechen. Hier soll es um die Unternehmens-gestützten Open Source-Lizenzen gehen, die seit Netscapes Schritt in die Bewegung hinein entstanden sind.

Die *Mozilla Public License* ist von der OSI als OSD-kompatibel lizenziert. Auf den zweiten Blick ist sie jedoch, ähnlich wie die *Not-for-profit Freely-redistributable* Lizenzen, wenn auch aus anderen Gründen, als 'halbfrei' einzuordnen. Die Open Source-Lizenz entstand, als Netscape sich Anfang 1998 entschloß, -- beraten von Raymond und Perens -- den Quellcode des Communicator 5.0 Standard Edition (bereinigt um allen Code, der geistiges Eigentum Dritter ist, und die Kryptografiemodule, die US-amerikanischen Ausfuhrbeschränkungen unterliegen) freizugeben. Die Codebasis wurde unter dem Namen Mozilla in ein freies Software-Projekt überführt. Der Lizenztext[387] trägt deutlicher als alle bislang behandelten Lizenzen die Handschrift von IP-Anwälten. Erstmals spricht eine freie Lizenz auch mögliche Patentansprüche an.[388] Als nach ausgiebigen Diskussion und öffentlicher Kommentierung ein Entwurf der Lizenz veröffentlicht wurde, richtete sich die Kritik vor allem gegen die Sonderrechte, die sich Netscape darin vorbehielt. Daraufhin entschloß sich das Unternehmen, zwei Lizenzen herauszugeben, die Netscape Public License 1.0 (NPL) für den am 31. März 1998 freigegebenen Communicator-Code und alle davon abgeleiteten Werke sowie die Mozilla Public License 1.0 (MPL), die Autoren für eigenständige Werke benutzen können, zu denen sie Netscape keinen privilegierten Zugang geben möchten. MPL und NPL bestehen aus einem identischen Hauptteil, dem bei der NPL die "*Amendments*" hinzugefügt sind, die den Sonderstatus von Netscape regeln.[389]

Die MPL gewährt gebührenfrei alle Freiheiten und verlangt, daß alle Modifikationen in Quellcodeform zugänglich gemacht und unter dieselbe Lizenz gestellt werden (Ziff. 3.2.). Ziff. 3.6. erlaubt allerdings, die veränderte oder unveränderte Software ausschließlich in Objektcodeversionen unter einer beliebigen anderen Lizenz zu verbreiten, sofern ein Hinweis auf den freien Quellcode beigefügt wird. Auch die Verbindung von MPL-Code mit Code unter anderer Lizenz zu einem *Larger Work* ist zugestanden (Ziff. 3.7).[390] Ein solches *Larger Work* wird nicht als abgeleitetes Werk interpretiert, kann also ebenfalls unter eine restriktivere Lizenz gestellt werden, solange der ursprüngliche und der davon abgeleitete Quellcode weiterhin von NPL oder MPL regiert werden. Damit wird eine Aufspaltung in die freie Code-Welt der Entwickler und in die der reinen Anwender erzeugt, denen alle

---

[387] http://www.mozilla.org/MPL/

[388] in den dunkleren 'legalesianischen' Passagen Ziff. 2.1.b und Ziff. 2.2.b. Die Annotationen erläutern, daß damit ausgeschlossen werden soll, daß Netscape oder andere Kontributoren Patentgebühren erheben könnten, andererseits sollen solche Patentrechte geschützt bleiben, "in a manner consistent with the goal of the License." (http://www.mozilla.org/MPL/annotated.html). Die beiden Ziffern, vor allem 2.2.b ist in der NPL/MPL 1.1 noch kryptischer geworden. Ferner wird die neue Ziff. 2.2.c eingefügt, derzufolge die beiden Kontributorenlizenzen (nach Copyright und nach Patentrecht) am Tag wirksam werden, da der Kontributor "*Commercial Use*" von seinem Code macht. Das aber ist verwirrenderweise als jegliche -- auch nicht-kommerzielle -- Verbreitung an Dritte definiert (Ziff. 1.0.1.). (Mozilla Public License Version 1.1, http://www.mozilla.org/MPL/MPL-1.1.html)

[389] Die derzeit gültige Version 1.1 faltet MPL & NPL in eins, ohne Datum, (Hinweis auf der Mantelseite: "Last modified September 24, 1999"), http://www.mozilla.org/MPL/MPL-1.1.html

[390] s.a. die Annotation zu 3.6. in http://www.mozilla.org/MPL/annotated.html

möglichen Restriktionen auferlegt werden können. Zwar werden aus der freien Quellcodebasis immer auch freie *Binaries* verfügbar sein, aber es ist leicht denkbar, daß ein Unternehmen wie Microsoft den freien Code um eigene attraktive Funktionalitäten erweitert und das Gesamtpaket ausschließlich in proprietärer Form verbreitet. Fühlen sich genug Nutzer von diesen Zusatzfunktionen angezogen und sind bereit, die Unfreiheit in kauf zu nehmen, verlieren die freien Entwickler ihre Anwender und das freie Projekt wird scheitern.

Außerdem nimmt die NPL/MPL eine ungewöhnliche Trennung in "*Initial Developer*" (für den NPL-Code ist das Netscape, für ein nicht-abgeleitetes eigenständiges Werk jeder Autor, der es unter die MPL stellt -- Ziff. 2.1) und "*Contributors*" (Ziff. 2.2) vor.

In den "*Amendments*" der NPL wird eine weitere Unterscheidung in die von Netscape *branded Versions* des Communicator und die freien Versionen unter dem Projektnamen Mozilla vorgenommen. Dort heißt es, daß Netscapes Warenzeichen (Namen und Logos) nicht mitlizenziert werden (Ziff. III.). Kontrovers sind die Abschnitte, die es Netscape erlauben, den NPL-Code einschließlich der Modifikationen durch Dritte in seinem *branded Code* (Ziff. V.3.) und im Laufe von zwei Jahren nach der ursprünglichen Freigabe von Mozilla auch in anderen Produkten (Ziff. V.2.) zu verwenden, ohne das es an seine eigene Lizenz gebunden ist. Netscape behält sich vor, Code unter der NPL unter anderen Bedingungen als der NPL an Dritte zu lizenzieren. Die Zusatzbestimmungen heben somit effektiv die Freiheitsvorschriften im Haupttext der NPL für die Firma Netscape wieder auf (Ziff. V.1.).

Zur Begründung hieß es, Netscape verwende einen Teil des Codes für den Client "Communicator" auch in seinen Server-Produkten und möchte sicherstellen, daß es Veränderungen am Server-Code vornehmen -- also auch Modifikationen von freien Entwicklern in den proprietären Code aufnehmen -- kann, ohne diesen gleichfalls unter die NPL stellen zu müssen. Außerdem hat Netscape Verträge mit Dritten über die Bereitstellung von Quellcode. Auch sie sollen von den Modifikationen freier Entwickler profitieren können, ohne ihre eigenen Erweiterungen unter der NPL zugänglich machen zu müssen.[391] Die Gefahr, die davon ausgeht, spielt Netscape jedoch herunter. Es re-lizenziere nur einen spezifischen *Snapshot* des Codes an einem bestimmten Datum. Wenn ein solcher Lizenznehmer nicht mit der Community zusammenarbeite, indem er seine eigenen Entwicklungen in die freie Codebasis zurückgibt, werde der von ihm lizenzierte Code mit der Zeit immer stärker von der freien 'Standardversion' abweichen.[392] Das, so wird suggeriert, sei eine hinreichende Motiviation für Hersteller, ihre Software freizugeben.

Das Argument unterschlägt, daß schon der Hautpteil der NPL (= MPL) mit Ziff. 3.7. "Larger Works" die Möglichkeit zuläßt, Mozilla oder Teile davon mit proprietären Modulen zu koppeln. Nach Ziff. 3.6 "Distribution of Executable Versions" würde dann ein Hinweis auf die Verfügbarkeit des Quellcodes der freien Bestandteile ausreichen und jeder kann sich aus dem freien Pool bedienen, Änderungen vornehmen und das Ergebnis zu Bedingungen der eigenen Wahl verwerten.

Freie Entwickler, die eine solche Privatisierung ausschließen wollen, können ihre eigenständigen Beiträge zum Mozilla-Code (nicht aber Modifikationen von NPL-Code)[393]

---

[391] Netscape Public License FAQ, 19.Will the NPL apply to Netscape as well?, http://www.mozilla.org/MPL/FAQ.html#19

[392] ebd.

[393] "if you add a new file that does not contain any of the Original Code or subsequent Modified code, it is not a Modification, and is not covered by the NPL", FAQ, op.cit. #4

unter die MPL (= NPL ohne *Amendments*) oder eine kompatible Lizenz, wie die BSD, die LGPL oder fast jede andere Lizenz stellen oder sie gar nicht veröffentlichen. Ausgeschlossen ist allein die GPL, von der Netscape behauptet, daß sie inkompatibel mit allen Lizenzen außer sich selbst sei.

> "Under our reading of the GPL, it will not be possible to incorporate code covered by the GPL into the Communicator source code base. It is also not possible to use GPLed code and NPLed code together in a Larger Work. This is different for LGPL code. It is possible to create a larger work using LGPLed code that can then be used in conjunction with NPLed code through an API."[394]

Diese Ansicht wird von der FSF bestätigt:

> "This [die MPL] is a free software license which is not a strong copyleft; unlike the X11 license, it has some complex restrictions that make it incompatible with the GNU GPL. That is, a module covered by the GPL and a module covered by the MPL cannot legally be linked together. We urge you not to use the MPL for this reason."[395]

Vergleichsweise harmlos, obgleich es ebenfalls viel Kritik auf sich gezogen hat, ist Netscapes Vorrecht, die Lizenz zu ändern. Denselben Mechanismus sieht auch die GPL vor. Keine Dynamik einzubauen wäre angesichts des raschen Wandels in Technologie und Recht auch unseriös. Doch selbst wenn Netscape oder gar die FSF plötzlich entscheiden sollten, MPL/NPL rsp. GPL in eine vollständig proprietarisierende Lizenz zu verwandeln, hätte das keine rückwirkende Folgen für alle vorangegangenen freien Versionen. Eine radikale Lizenzänderung würde allerdings unweigerlich eine Split in der weiteren Entwicklung der Code-Basis auslösen.[396]

Es ist hier nicht der Ort, um über die Hintergründe von Netscapes Business-Entscheidung zu spekulieren. Die veröffentlichten Motive sind, "to balance our goals of engaging the free source development community and continuing to meet our business objectives. ... The NPL and MozPL both attempt to strike a middle ground between promoting free source development by commercial enterprises and protecting free source developers."[397]

Netscape wollte also nicht die Tore zu seiner gesamte Produktlinie öffnen,[398] sondern nur zu Mozilla, d.h. dem bereinigten Code des Communicator 5.0 Standard Edition. Netscape schenkte der freien Software-Community einen Web-Browser und wollte -- quasi als Gegenleistung -- die 'freie' (im Doppelsinn von 'freiwilliger' und 'unbezahlter') Zuarbeit proprietär verwerten. "Netscape believes that it is in the interest of all who develop on the

---

[394] MPL FAQ #18

[395] http://www.gnu.org/philosophy/license-list.html

[396] vgl. NPL FAQ #24

[397] NPL FAQ #4 und #9

[398] Obgleich Netscape im FAQ die Absicht kundtut, auch seine Server freizugeben. Dort heißt es: "22.Why do you have a two year time limit in amendment V.2 of the NPL? This section is here because of the state of Netscape's code today. Netscape has a number of server products that contain NPL code. We intend to make sure those products can comply with the terms of NPL license as soon as practical." (http://www.mozilla.org/MPL/FAQ.html#22)

Communicator code base to contribute their changes back to the development community. In this way, they will reap the benefits of distributed development."[399] Viele in der Diskussion damals unterstellen Netscape, daß es solche Aussagen ehrlich meinte, doch spätestens als AOL Netscape (für $ 4 Mrd.) kaufte, kamen lautstarke Zweifel auf. Entscheidend sind nicht solche Aussagen, sondern der lizenzrechtliche Status des gemeinsam bearbeiteten und genutzten Codes. Zwar wies Jamie Zawinski, im ersten Jahr mozilla.org Core-Team-Mitglied, darauf hin, daß die Freiheiten, die Mozilla durch die NPL gewährt wurden, nicht nachträglich entfernt werden können[400] -- einmal aus der Copyright-Flasche, kann kann der Code nicht zurückgerufen werden. Doch die weitere Pflege und Entwicklung des Codes hängt mit der NPL/MPL zu einem gewissen Maß von der Benevolenz ihres industriellen Hauptnutzers ab.

Netscapes Lizenzmodell hat eine Reihe Nachfolger gefunden. In der Regel verwenden sie MPL, also ohne den Sonderstatus, den die NPL dem ausgebenden Unternehmen verleiht. Dazu gehört Cygnus, das sein *embedded Cygnus operating system* (eCos), im Oktober 1998 released, unter die Cygnus eCos Public License ver 1.0 gestellt hat, die wortgleich zur MPL 1.0 ist, nur das 'Netscape' gegen 'Cygnus', und, nachdem Cygnus an Red Hat verkauft worden ist, gegen 'Red Hat' ausgetauscht wurde.[401]

Ebenso verfuhr Ricoh Silicon Valley, Inc. mit seiner Ricoh Source Code Public License,[402] die es für seine *Platform for Information Applications* (PIA) verwendet, eine Umgebung für die rasche Entwicklung von leicht zu unterhaltenden Web-basierten Anwendungen.

Open Source hat auch bereits Eingang in die Hochfinanz gehalten. Den Originaltext der MPL ver. 1.0, einschließlich Netscapes Vorrecht, die Lizenz zu ändern, verwendet Price-Waterhouse für sein FpML™ (Financial products Markup Language),[403] ein XML-basiertes Protokoll für Business-to-Business eCommerce im Bereich von Finanzderivativen.

Die *Netizen Open Source License* (NOSL), Version 1.0.[404] beruht auf der MPL 1.1, ersetzt nur 'Netscape' durch 'Netizen Pty Ltd.' und fügt einige Australien-spezifische Bits ein. Es ist die Lizenz von Eureka, einem Satz von Data-Mining-Tools für HTTP-Log-Dateien. Wie die MPL hat sie eine Fassung mit Zusätzen, die die hostende Firma favorisieren, in diesem Fall Netizen, eine 15-Personen Firma, die sich auf Consulting, Entwicklung und Training für Open Source-Software und Internet-Technologie spezialisert hat. Ein zweites Produkt von Netizen, das Projekt/Task Management-System Xen, steht unter der *Xen Open Source3 License* Version 1.0 (XOSL), also der NOSL mit denselben Zusätzen wie die NPL.

Auch Sun Microsystems hat neben andere Linzenzmodellen MPL-Varianten in Verwendung. Bei der *Sun Public License ver 1.0*[405] für NetBeans, eine in Java geschriebene integrierte Entwicklerumgebung, handelt es sich um den Text der MLP 1.1, bei der der Freiheit des Code auch die der Dokumentation hinzugefügt und die Markennamen ausgetauscht wurden.

---

[399] NPL FAQ #19

[400] Zawinski 1998

[401] http://sourceware.redhat.com/ecos/license.html

[402] http://www.risource.org/RPL/RPL-1.0A.shtml

[403] http://www.fpml.org/documents/license/index.html

[404] http://bits.netizen.com.au/licenses/NOSL/

[405] http://www.netbeans.org/license.html

Eine andere Strategie verwendet Sun für sein NFS (Network Filesharing System). Die *Sun Industry Standards Source License 1.0* (SISSL)[406] besteht zu großen Teilen aus der MPL 1.1. Die Ziffern 2.2 bis 3.4 der MPL, die sich auf die Modifikationen durch *Contributors* im Gegensatz zum *Initial Developer* beziehen, entfallen. Sun bindet mit der SISSL vor allem den Originalcode, also seinen eigenen. Die 'Auslieferung' einer 'Kontributorenversion' unterstellt sie einer besonderen Auflage. 120 Tage vorher muß sie die Anforderungen eines Standardisierungsgremiums erfüllen (Ziff. 3.0), was mit Hilfe eines Kompatibilitäts-Kits getestet wird. Nur für den Fall, daß die geänderte Version vom Standard abweicht, verpflichtet die SISSL den Autor, die Dokumentation und eine Referenzimplementation seiner Software unter denselben Bedingungen wie die der SISSL öffentlich zu machen. Wer die Standards einhält, kann also Modifikationen auch proprietär halten. Suns erklärtes Ziel ist es, die NFS-Infrastruktur zu stärken, deren Spezifikationen es als öffentliche Standards an die IETF übertragen hat.[407]

Für Java und Jini führte Sun im Januar 1999 eine dritte Lizenz ein, die *Sun Community Source License* (SCSL),[408] die eigentlich drei Lizenzen in einer sind. In kumulativer Abstufung enthält sie die 1) *Research Use License*, die für Evaluation, Forschung, Entwicklung und Prototyping potentieller Produkte die größten Freiheiten gewährt, die 2) *Internal Deployment Use License* für eine sehr begrenzte interne Distribution, um Produkte vor ihrer Markteinführung zu testen, und schließlich die 3) *Commercial Use License*, die jeder unterzeichnen muß, der Original und Modifikationen verbreiten möchte. Nach 1) darf Quellcode einschließlich Modifikationen nur an andere Lizenznehmer verbreitet werden. Nach 2) muß jeder Code, wie bei der SISSL, den Test mit dem *Java Compatibility Kit* (JCK) bestehen. Wie bei NSF handelt es sich bei Java und Jini um infrastrukturelle Technologien, mit einem Kern, den Sun unter enger Kontrolle hält, und einer Vielzahl möglicher Diensten drumherum, die von möglichst vielen ausgebaut werden müssen, damit der Wert des Systems für alle Beteiligten wächst. Um Javas Cross-Plattform-Versprechen "write once, run anywhere" halten zu können, muß Sun dafür sorgen, daß in allen Implemententationen ein für alle verbindlicher Standard eingehalten wird. Das Anliegen ist umso verständlicher, als Micrsoft einen schwergewichtigen Versuch unternommen hat, Java auf Kompatibilitäts-brechende Weise zu verändern. Die beiden Unternehmen standen sich deshalb gerade vor Gericht gegenüber, als die SCSL verfaßt wurde. Zur Wahrung der Einhaltung der Standards dienen zwei Mechanismen: Arbeitsausschüsse der beteiligten Organsiationen und der JCK. Der JCK ist ein komplexes Werkzeug, dessen Benutzung an eine eigene Lizenz und vor allem -- da er so komplex ist -- an einen kostspieligen Support-Vertrag mit Sun oder zertifizierten Dritten gebunden ist.[409] Das sind nun keine Anforderungen mit denen die freie Software-Welt üblicherweise zu tun hat. Mike Loukides gesteht Sun, bei aller Kritik an der SCSL, zu, daß es mit dieser Konstruktion auch auf die Angriffe von Microsoft antworte. "It's not a problem that the open source community has faced, and it's unclear how a completely open process would respond. What would happen,

---

[406] http://soldc.sun.com/SIndStanSrcLic_012800.pdf

[407] Suns ISSL-FAQ, http://soldc.sun.com/tools/licensefaq2.html (z.T. wörtlich aus dem von Netscape übernommen)

[408] die es in mehreren, geringfügig voneinander abweichenden Versionen gibt, die Jini-SCSL ist http://www.sun.com/jini/licensing/scsl_jcp_v.1.6c_web.html

[409] Sun's New Java™ Source Licencing Policies, an Interview with George Paolini, Vice President of Marketing, Sun Java Software, http://java.sun.com/features/1998/12/source_license_QA.html

for example, if Microsoft decided that it was in their interest to introduce incompatibilities into Perl, and herd their developers towards some private version? I think this will happen -- probably sooner rather than later."[410]

Sun vermeidet sorgfältig, die SCSL als eine 'Open Source'-Lizenz zu bezeichnen. Weder sie noch die SISSL ist von der OSI als OSD-gemäß zertifiziert worden. Und auch die FSF identifiziert beide als inkompatibel mit der GPL.

Wie Netscape möchte Sun das Beste aus der Welt der proprietären und der Open Source-Lizenzen vereinigen. Als Vorteile von letzteren sehen Gabriel und Joy eine verstärkte Innovation, die Vergrößerung der effektiven Arbeiterschaft, verbesserte Qualtität und eine schnellere Kommerzialisierung. "A participating organization can reap the benefits of expertise not in its employ."[411] Auch der kommerzielle Vorteil ist deutlich. Sun wird erste Adresse für Java und Jini bleiben. "Thus, even when source is out in the open, the value still remains with those who can expertly manipulate it." Und auch von den Lizenznehmern fließen Einnahmen zurück, nicht mehr, wie bislang, vor Beginn der Entwicklung, sondern in dem Moment, wo auch der Lizenznehmer Geld verdient.[412]

Die 'Community', die die SCSL vorsieht und schafft, besteht aus einer 'entwickelnden Organisation' und hinzukommenden Organisationen. Freie Entwickler tauchen nicht auf, sind allenfalls als Lehrende und Studierende hochwillkommen. Wo die freien Lizenzen ein Netzwerk von Individuen im Auge haben, zielt die SCSL auf ein Netzwerk von Firmen. "For example, Jini technology licensing is based on a community of companies who wish to build and sell Jini devices. Sun Microsystems is the developing organization which invented, designed, and built the initial Jini infrastructure."[413]

Auch andere Softwareunternehmen entfernen sich stärker von Netscapes Lizenzmodell. So hat IBM seinen Jikes-Compiler unter die *IBM Public License 1.0*[414] gestellt, die der MPL ähnelt und von der OSI als OSD-kompatibel zertifiziert worden ist. Kontroverser wurde die *Apple Public Source License* (APSL)[415] aufgenommen, die u.a. Darwin (den Kern von Mac OS X) und den Darwin Streaming Server (einschließlich QuickTime-Streaming) für registrierte Entwickler zugänglich macht. Mit ihrem Sonderstatus für das ausgebende Unternehmen ähnelt sie der NPL. Kontributoren müssen Apple eine unwiederrufliche Lizenz erteilen, die Copyright- und Patentrechte an ihren Modifikationen zu nutzen (Ziff 3), gleichzeitig behält sich Apple vor, die APSL im Falle von Patentstreitigkeiten pauschal zu wiederrufen (Ziff. 9.1.c). Modifikationen der Kontributoren müssen im Quellcode freigegeben und mit einem Online-Formular bei Apple angemeldet werden (Ziff 2.2.c), gleichzeitig behält sich Apple alle Rechte am Originalcode und an seinen eigenen Modifikationen vor und entbindet sich selbst bei deren Nutzung von seiner eigenen Lizenz (Ziff 11). Im März 1999 verurteilten Perens (der inzwischen die OSI verlassen und die Seite der FSF ergriffen hatte) und andere in einem offenen Brief[416] die Entscheidung der OSI, der APSL das OSD-Gütesiegel zu verleihen. U.a. wiesen sie darauf hin, daß erhebliche

---

[410] Loukides 12/1999

[411] Gabriel/Joy 1999

[412] Paolini 1998

[413] Gabriel/Joy 1999

[414] http://www.research.ibm.com/jikes/license/license3.html

[415] The Apple Public Source License 1.1, 19. April 1999, http://www.publicsource.apple.com/apsl/

[416] Bruce Perens, Wichert Akkerman, Ian Jackson, The Apple Public Source License - Our Concerns, 16 March 1999, http://www.perens.com/APSL.html

Teile des von Apple unter der APSL -- und eigenem Copyright -- veröffentlichten Codes unwesentliche Modifikationen von Code aus der Berkeley Universität und der Carnegie-Mellon seien. Diese sind vom Steuerzahler finanziert und stehen unter freien Lizenzen und sollten deshalb von der APSL verschont bleiben. Raymond verteidigte die Entscheidung zunächst,[417] doch schließlich wurde die Zertifizierung der APSL zurückgenommen.

Die Lizenzkonstruktionen von Netscape, Sun, IBM und Apple weichen deutlich vom herkömmlichen proprietären Modell ab. Hier sei an den Unterschied zwischen 'kommerzieller' und 'proprietärer' Nutzung erinnert. Kommerzielle Software wir von Firmen entwickelt, die damit Geld verdienen wollen. Sie ist in der Regel proprietär, aber es gibt auch kommerzielle Software unter GPL (z.B. GNU Ada) und ebenso nichtkommerzielle proprietäre Software. Strenggenommen haben auch GPL-Programme einen *proprietas*, einen Eigentümer, nämlich den oder die jeweiligen Copyright-Halter. Und auch die FSF hat keine Einwände dagegen, mit freier Software Geld zu verdienen. Der Unterschied liegt in der Gewichtung. Priorität der FSF sind die Freiheiten. Priorität der genannten Unternehmen ist ein, wenn auch ungewöhnliches, so doch unzweifelhaft auf Profitmaximierung zielendes Business-Modell.

Beide gleichen sich darin, daß sie eine geschlossene Gemeinschaft von Entwicklern erzeugen, die Quellcode miteineinander teilen und sich zu gegenseitiger Kooperation verpflichten. Bei den Kommerziellen wird man Mitglied in dieser In-Group durch einen schlichten Mausklick auf den 'Akzeptieren'-Knopf der Lizenz und meist eine Registrierung. Unter den Kooperierenden gibt es hier eine bevorrechtigte Partei. Sun beispielsweise wird als Eigentümerin von Plattformtechnologie von deren Gedeihen in jedem Fall profitieren, selbst wenn es keine Lizenzzahlungen bekommen sollte.

Für solche geschlossenen Gemeinschaft ist der Begriff '*Gated Communities*' geprägt worden. Tim O'Reilly erläutert ihn an einem Beispiel. Sein Unternehmen verwendet ein Software-Paket für Verlage namens CISpub, ein proprietäres Produkt mit höchstens einigen hundert Nutzern, die aber alle Zugriff auf den Quellcode haben und es aktiv weiterentwickeln. Der freie Austausch ist auf die Nutzerbasis beschränkt, und um Nutzer zu werden muß man CISpub käuflich erwerben. O'Reilly hält es auch für unwahrscheinlich, daß sich irgendjemand anderes dafür interessieren könnte. "This is a specialized package written in a specialized language for a specialized industry segment."[418] Den wichtigsten Vorteil sieht O'Reilly darin, daß umzäunte Communities einen Weg darstellen, auf dem Open Source-Ethik und Methodologie in die Welt der spezialisierten Businesswelt, zu denjenigen, die noch nicht bereit sind, vollständig den Open Source-Weg zu beschreiten, vordringen könne. Im übrigen schließe das Modell an die frühen Tage von Unix und IBM-Software an, in denen ebenfalls Lizenznehmer reichlich Code entwickelten und mit der Nutzerbasis teilten.

> "In short, if a 'gated source community' means that I can share my changes only with other existing licensees of the base package, but not with any outsiders, I'm still for it. [...] To make this work, you need to use licensing terms allowing redistribution of modifications to other licensees, public repositories for contributed code, discussion

[417] Eric S. Raymond, OSI clarifies the status of the APSL, 17 Mar 1999, http://www.perens.com/APSL.html

[418] O'Reilly 5/2000

> forums so users can find each other, and other mechanisms familiar to open source developers.[419] It also helps if you have a modular architecture that makes it easier for people to add-on without changing more than they need to. [...] Perhaps the term 'gated source community' has some negative connotations, since it suggests exclusivity, but at bottom, all it means is enabling users of a particular piece of software to form their own private community for source code access and sharing. This is a very appealing scenario for vendors and customers alike."[420]

O'Reilly nimmt dem negativen Beiklang weiterhin die Spitze, indem er auf die GPL-Community verweist.

> "Incidentally, you could argue that the GPL (though not BSD-style licenses) creates a kind of gated community, in that only those agreeing to the license can (in theory) redistribute the source code. Because of the terms of the license, this is an evangelical gated community, always trying to bring in new members, but it's still closed to those who don't want to abide by the license agreement."[421]

Dem ist zuzustimmen. Beide Modelle stecken Territorien von Privateigentum und Kollektiveigentum ab. Gabriel/Joy sehen den Vorteil des Open Source-Modells (das, anders als 'freie Software' keine "politischen Überzeugungen" darüber verkörpere, was Eigentum sein kann und was nicht) u.a. darin, daß "[t]here is a self-organizing effect in which the boundaries between proprietary concerns and community concerns are adaptively set."[422] Diese Selbstorganisation nimmt in den Lizenzen Form an, was es umso plausibler erscheinen läßt, daß Debian seine Lizenz einen 'Gesellschaftsvertrag' nennt.

Stallman schreibt (in seiner Kritik an der APSL): "[T]he spirit of free software, which is that we form a community to cooperate on the commons of software."[423] Diese Idee eines *Commons*, zu Deutsch 'Allmende', oder genauer Wissens-Allmende, ist beim GNU-Projekt am konsequentesten durchdacht, das darauf zielt, einen vollständigen freien Software-Korpus aufzubauen, der erlaubt, alles mit dem Computer zu machen. Wissen braucht, ebenso wie natürliche Ressourcen, Pflege und Weiterentwicklung, also Investitionen. Die werden im Falle einer Allmende von den Allmendgenossen getragen, hier der Community freier Entwickler. Zur Erhaltung und Pflege der gemeinsamen Investitionen werden bestimmte Nutzungen ausgeschlossen. Deshalb steht GNU-Software ausdrücklich nicht in der *public Domain*, sondern schöpft den Eigentumsanspruch des Copyright voll aus. Auch das GNU-Projekt ist also ein geschlossener Wissensraum, eine *gated Community*. Ihre Grenzen werden von der GPL abgesteckt. Die Vereinbarungen, die die Almendgenossen unter sich treffen, nehmen bei Wissensgütern die Form von Lizenzen an. Wer sie akzeptiert und respektiert

---

[419] Collab.net hat sich, neben dem Hosting von Open Source-Projekten, auf solche umzäunten Gemeinschaften spezialisiert.

[420] O'Reilly 2000

[421] ebd.

[422] Gabriel/Joy 1/1999

[423] Richard Stallman, The Problems of the Apple License, 1999, http://www.gnu.org/philosophy/apsl.html. Ebenso Moglen über die GPL: "This use of intellectual property rules to create a commons in cyberspace is the central institutional structure enabling the anarchist triumph." Moglen 1999

gehört dazu. Wer gegen sie verstößt wird automatisch von der Nutzung ausgeschlossen (die Verfallsklausel Ziff. 4 GPL). Die GPL ist der Maßstab, der eine Lizenz genügen muß, damit eine Software in das GNU-Projekt aufgenommen werden kann -- die Grenze ins Innere der Allmende passieren darf.

Der *Uniform Computer Information Transactions Act*

Vollkommen unfreie Lizenzen, wie die konventionelle Software-Industrie sie verwendet, verbieten die Nutzung durch Dritte, das Kopieren, die Weiterverbreitung und die Modifikation. Sie lizenziert binäre, also ausschließlich ausführbare, nicht aber veränderbare Versionen. In einigen Fällen wird auch der Quellcode angeboten, gegen zusätzliche Gebühren und nur für beschränkte Zwecke. Netscapes *Tools End User License Agreement*[424] und Microsofts *End User License Agreement* (EULA) sind Beispiele für solche Lizenzen. Wer nachlesen möchte, wie Unternehmen von Sega über Microsoft und AT&T bis Caldera ihr geistiges Eigentum untereinander lizenzieren wird auf "*Tech Deals: Intellectual Property Licenses*"[425] reiche Beute finden.

Um es noch einmal deutlich zu sagen: Unternehmen verkaufen ihre Software (Bücher, Musik-CDs, Video-DVDs, MP3-Dateien, usw.) nicht an die Endkunden, sondern lizenzieren nur bestimmte Nutzungsrechte daran. Und selbst diese werden immer stärker eingeschränkt. Microsoft verkündete zum 1. Januar 2000 eine Innovation in seiner Lizenzpolitik, derzufolge künftige Versionen von MS-Windows nicht mehr frei re-installierbar sind.[426] *Original Equimpment Manufacturers* (OEMs) dürfen seither keine vollwertigen Windows-CDs mit ihrer Hardware ausliefern. Vielmehr spielen sie ein *Disk-Image* des Betriebssystems auf die Festplatte und geben den Kunden eine "*Recovery CD*" mit, die ausschließlich auf dem jeweiligen Computer läuft. Ist das vorinstallierte Windows zerschossen oder die Festplatte versagt oder der Kunde ersetzt sie durch eine neue, so kann er die Software von dieser CD neu installieren. Eine vollwertige Version erlaubt es, Windows über eine defekte Installation erneut zu installieren -- keine elegante, aber eine in vielen Fällen wirksame Methode, die zudem andere installierte Software und individuelle Konfigurationen unberührt läßt. Die "Recovery CD" dagegen überschreibt alle Windows-Dateien und damit die *Registry* für andere Programme, alle Konfigurationen und Updates seit der Erstinstallation. Tauscht der Kunde die Grundplatine mit dem BIOS-ROM aus, dessen Kennung bei dieser 'Individualisierung' abgefragt wird, so verliert er vollständig die Möglichkeit, von seinen für etwa 200 DM erworbenen Nutzungsrechten Gebrauch zu machen. Jeder OEM, wie Compaq, Toshiba und Dell usw., fügt dem BIOS eine individuelle Kennung bei, die von der Recovery-CD abgefragt wird und verhindert, daß sie auf einem Rechner eines anderen Herstellers verwendet werden kann. Auf einem anderen Rechner desselben Herstellers mit derselben BIOS-Signatur ließe sie sich verwenden, doch das ist eine hypothetische Option, da Microsoft es in seinen Verträgen mit den OEMs untersagt, Rechner ohne ein vorinstalliertes

[424] http://developer.netscape.com/software/jsdebug_license.html

[425] http://techdeals.findlaw.com/license/index.html

[426] Microsoft trägt sich schon seit einigen Jahren mit dem Plan, Software überhaupt nicht mehr zeitlich unbegrenzt zu lizenzieren, sondern nur noch zu vermieten. Aufgrund des Kartellverfahrens hat es davon bislang noch Abstand genommen.

MS-Betriebssystem zu vekaufen. Dieses *BIOS Lock* soll Software-'Piraterie' verhindern.[427] Zwar war es auch nach bisheriger Lizenzpolitik nicht zulässig, ein Windows auf zwei Rechnern zu installieren, doch es auf dem ersten zu löschen und dann auf dem zweiten zu installieren, war rechtens und sollte es auch bleiben, nur daß Microsofts Stragie es technisch verunmöglicht.

Dieser Trend zur 'Individualisierung' von Software und digitalen Inhalten zeigt sich auch an anderen Stellen. Auch MS-Office 2000 geht durch eine Zwangregistrierung eine ähnliche Bindung mit dem jeweiligen System ein, wie die Recovery-CD bei der Installation durch den OEM. Ähnliches plant auch die Musikindustrie, die uns Stücke verkaufen möchte, die nur auf einem einzigen Abspielgerät gehört werden können.

Nach der Erstverkaufsdoktrin von Copyright und Urheberrecht darf der rechtmäßige Besitzer eines Buches, einer Musik-CD usw. dieses Werkstück (nicht aber Kopien davon) weiterverkaufen. Was das Recht auch für Software erlaubt, verbieten die Lizenzen, vor allem für OEM-Software ("Vertrieb nur mit einem PC", "Produktunterstützung erhalten Sie vom Hersteller des PCs"), die lizenztechnisch an die Hardware gekoppelt ist. Sie wird nur zusammen mit der Hardware verkauft und darf ausschließlich mit ihr zusammen weiterverkauft werden, und auch dann nur, wenn alle Sicherungskopien gelöscht sind. Wer z.B. Linux auf dem betreffenden Rehner installiert, darf also nicht einen Teil der Investitionen in die MS-Software zurückgewinnen, indem er sie verkauft. Ein Markt für gebrauchte Software wird so unterbunden, und ebenso, daß die Kunden von ihrem Weiterverkaufsrecht Gebrauch machen.

Bis 1999 die großen OEMs begannen, auch Rechner mit vorinstalliertem Linux anzubieten, gab es für einen Käufer nur einen Weg, an Microsoft vorbeizukommen: die Lizenz zu verweigern. Wer bei "Akzeptieren?" auf "Nein" klickt, dem wird die Nutzung der MS-Software verweigert und mitgeteilt, daß er sein Windows gegen Rückerstattung des Kaufpreises zu seinem Händler zurückbringen kann.[428] Doch auch diese Option war eine rein hypothetische, bis Ende 1998 ein australischer Linux-Nutzer nach hartnäckigen und langwierigen Bemühungen, bei denen sich Microsoft, OEM und Händler gegenseitig die Verantwortung zuschoben, tatsächlich erstmals eine Rückerstattung erwirken konnte.

Das alles hat mit Copyright/Urheberrecht wenig zu tun. Beide geben dem Konsumenten das Recht, Sicherheitskopien anzufertigen, Sofware zu dekompilieren, um Fehler zu beheben und interoperable Programme zu erstellen, und ein nicht mehr benötigtes Programm an Dritte weiterzuverkaufen -- Rechte, die ihm die Lizenz verweigern. Einem Autor gibt das Gesetz das grundsätzliche Recht, sein Werk zu Bedinungen seiner Wahl zu veröffentlichen, solange diese nicht gegen geltendes Recht verstoßen. Genaugenommen ist ein Urheberrechtsanspruch nicht einmal erforderlich. Eine Firma könnte selbst gemeinfreies Material unter einer restriktiven Lizenz verkaufen, solange ihre Kunden bereit sind, sie zu akzeptieren. Bei den genannten Mechanismen wird das Urheberrecht mit seiner lästigen Balance zwischen Rechteinhaber und Öffentlichkeit gleichsam links liegen gelassen, während die möglichen Nutzungen von Lizenz und Technologie regiert werden. Wo z.B. das Recht (nach der Erstverkaufsdoktrin) die Möglichkeit vorsieht, Werkstücke weiterzuverkaufen, verhindern Lizenz und Technologie dies.

---

[427] zur Technologie von Bios-Lock und Recovery-CD vgl. die Information von @priori, einem Dritthersteller für die neue Microsoft-Anforderung, http://www.at-priori.com/ und /faq.htm

[428] "Falls Sie den Bestimmungen dieses Lizenzvertrags nicht zustimmen, geben Sie bitte das unbenutzte Produkt unverzüglich gegen Rückerstattung des Kaufpreises zurück." (MS-Windows 98)

Die Gültigkeit von sog. Massenmarktlizenzen (*Shrink-Wrap Licences*, die binded werden, sobald der Kunde die verschweißte Packung öffnet, oder *Click-Through Licenses* für die Online-Distribution, die mit dem Anklicken eines "Lizenz akzeptieren"-Knopfes bindend werden; nach demselben Mechanismus sehen auch die Lizenzen der freien Software vor, daß der Nutzer durch Verbreitung oder Veränderung des Programms seine Einwilligung in die Lizenzbedingungen anzeigt) ist umstritten. So schreib Siepmann für die deutsche Rechtslage:"AGB auf Schutzhüllen von Datenträgern (sog. 'Shrink-Wrap-Agreements') haben aus vertragsrechtlicher Sicht im allgemeinen keine Gültigkeit, da diese erst nach Vertragsschluß zur Kenntnis genommen werden können. Sie können jedoch urheberrechtlich von Bedeutung sein."[429] Lizenzen waren individuell zwischen Firmen ausgehandelte und unterzeichnete Verträge, bis mit dem PC ein anonmyer Massenmarkt für Software aufkam. Für diesen Bereich enwickelten die 'Inhaltsbesitzer' das vereinfachte Lizenzierungsverfahren. Doch viele amerikanische Richter weigern sich bislang, die *Shrink-Wrap*-Lizenzen durchzusetzen. Mit der Bezahlung der Ware im Laden, so die Argumentation, sei ein Kaufvertrag zustande gekommen. Die Lizenz, die der Käufer erst zur Kenntnis nehmen kann, wenn er die Packung öffnet, sei ein Versuch, die Natur der Transaktion durch zusätzliche Bedingungen nachträglich zu verändern. Diesen geänderten Vertragsbedingungen muß der Käufer separat zustimmen, und dafür reiche ein Mausklick nicht aus.

Diese Rechtsunsicherheit sollte im Zuge der Revision der US-amerikanischen *Uniform Commercial Code* (UCC), dem Äquivalent zu den deutschen Allgemeinen Geschäftsbedingungen (AGB), beseitigt werden. Zur Begründung heißt es: "As the nation moves from an economy centered around transactions in goods and services to an information economy, the need has grown dramatically for coherent and predictable legal rules to support the contracts that underlie that economy. Lack of uniformity and lack of clarity of the legal rules governing these transactions engender uncertainty, unpredictability, and high transaction costs."[430]

Die Reform des UCC wurde gemeinsam vom *American Law Institute* (ALI) und der *National Conference of Commissioners on Uniform State Laws* (NCCUSL) betrieben, da Vertragsrecht Ländersache ist. Anfangs waren Bestimmung zu Computer-Programmen als Artikel 2b des UCC geplant, wobei das Gesetz aber ansonsten den Handel mit materiellen Gütern behandelt. Mitte 1999 gaben die Beteiligten bekannt, daß die Regeln für Computer-Informationstransaktionen in einem eigenständige Rahmengesetz, dem *Uniform Computer Information Transactions Act* (UCITA)[431] geregelt werden, das derzeit in den einzelnen US-Bundesländern umgesetzt wird.[432]

Das UCITA legalisiert *Shrink-Wrap*- (Ziff. 209) und Online-Lizenzen (Ziff. 211) für die Nutzung von 'Computer-Information' (nicht nur Programme, sondern jede Art elektronischer Inhalte, die von einem Computer verarbeitet werden können, einschließlich

---

[429] Siepmann 1999: Abs. 53

[430] ALI und NCCUSL, Presseerklärung, April 7, 1999, http://www.law.upenn.edu/bll/ulc/ucita/2brel.htm

[431] Die gültige Fassung ist die vom 9. Februar 2000, entworfen im Juli 1999, http://www.law.upenn.edu/bll/ulc/ucita/ucita200.htm. Der offizielle Kommentar dazu: http://www.law.upenn.edu/bll/ulc/ucita/ucitacom300.htm. Zu Entwurfsgeschichte, Revisionen und offizieller Kommentierung s. http://www.law.upenn.edu/bll/ulc/ulc.htm#ucita

[432] Maryland und Iowa haben bereits entsprechende Gesetze erlassen. Für Implementierungs-Updates s. http://www.ucitaonline.com/whathap.html

der dazugehörigen Dokumentation -- Ziff. 102.10), sofern der Lizenznehmer die Möglichkeit hat, die Vertragsbedingungen zur Kenntnis zu nehmen, bevor er seine Zustimmung manifestieren muß. Daneben regelt das UCITA den Zugang für eine bestimmte Zeitspanne zu Online-Informationen (Ziff. 611).

Besonders umstritten ist der Passus, der es Software-Herstellern erlaubt, Mechanismen zur *Electronic Self-Help Repossession* in ihre Produkte einzubauen, die im Falle eines Vertragsbruchs durch den Lizenznehmer ausgelöst werden können (Ziff. 816). Mit '*Self-Help*' ist gemeint, daß das Unternehmen bei einem (tatsächlichen oder vermeintlichen) Verstoß des Lizenznehmers, ohne eine Gericht anzurufen, die Lizenzenz wiederrufen und 15 Tage nach einer Vorwarnung,[433] mit elektronischen Mitteln,[434] z.B. über das Internet, die Programmversion des Kunden deaktivieren oder löschen kann. Die 'Selbsthilfe' der Industrie ist an einige Bedingungen gebunden (der Abschnitt ist auch nicht 'Self-Help', sondern "Limitations on Electronic Self-Help" betitelt), doch grundsätzlich problematisch an dieser 'Wiederaneignung' bleibt, daß das Rechtsgut der geschützten Privatsphäre (der Festplatte des Nutzers) im Interesse des Rechts von Copyright-Eigentümern, die Nutzung ihrer Werke zu kontrollieren, eingeschränkt wird.

Auch ein Verbot auf den Wiederverkauf von Massenmarktlizenzen, sofern es es deutlich kenntlich gemacht wird, legalisiert das UCITA (Ziff. 503.4). Tatsächlich liefert Ziff. 503.1(b) ("A party's contractual interest may be transferred unless the transfer ... would ... materially impair the other party's property or its likelihood or expectation of obtaining return performance.") das Argument dafür, jeglichen Second-Hand-Markt für digitales Wissen zu unterbinden.

Befürworter des UCITA sehen es als einen Fortschritt im Konsumentenschutz. Zwar gewährt es (den Inhaltsanbietern) weitgehende Vertragsfreiheit, aber es schreibt auch einen Minimalsatz von Rechten fest, auf deren Verzicht keine (für den Konsumenten nicht-verhandelbare) Lizenz die Vertragsparteien festschreiben kann (Ziff. 113), darunter explizite und implizite Garantieansprüche (Teil 4), z.B. die -- wenn auch mehrfach eingeschränkte -- Garantie, daß die gelieferte Information das hält, was der Anbieter in Werbung oder Demonstrationen versprochen hat.

Interessanterweise betont Carol Kunze auf der nichtoffiziellen Ucitaonline.com[435] unter dem Titel "Myths about UCITA", daß es gerade nicht verhindere, daß Hersteller alle Garantieansprüche ausschließen. Das ist tatsächlich übliche Praxis in der Software-Branche. Die Rechtslage ändere sich nicht. Software kann "*as is*" verkauft werden. Die Belehrung ist direkt an die Linux-Anhänger adressiert, die 'das traurigste Beispiel' für häufige Mißverständnisse des UCITA abgeben würden.[436] Ihre Ablehnung begründe sich darin, daß die Linux-Anhänger gesagt bekommen hätten, daß das UCITA den Lizenzgebern erlaube, sich von allen Garantieansprüchen freizusprechen, was zur Auslieferung von defekten Produkten führen würde. Die absurde Mißrepräsentation der Kritik der freien Software-Welt benutzt Kunze, um sie zu wiederlegen. Diese, sagt sie -- zu recht --, habe gerade kein

---

[433] selbst diese Auflage scheint durch Ziff. 816(i) wieder aufgehoben zu werden.

[434] In der Debatte war von "reasonably configured electronic agents" die Rede. Im offiziellen Kommentar heißt es dazu: "Prior law on use of electronic measures to enforce remedies on breach is unclear." http://www.law.upenn.edu/bll/ulc/ucita/ucitacom300.htm

[435] vormals unter http://www.SoftwareIndustry.org

[436] Kunze empfiehlt ihnen, daß sie sich einen Anwalt nehmen, bevor sie sich selbst schaden, und stellt zugleich seinen Mangel an Verständnis der freien Software unter Beweis (MS-IE als freie Software).

Interesse an rechtlichen Garantieansprüchen. Es sei ja gerade die Möglichkeit des Garantieausschlusses, die den Aufstieg der freien Software hervorgebracht habe [sic!]. Auch die verbreitete Fehlersuchtechnik der Betaversionen würde verschwinden, wenn man eine Garantie vorschreiben wollte.[437]

Die juristische Untermauerung für diese vertrackte Logik lieferte Robert Gomulkiewicz im Houston Law Review unter dem Titel "How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B".[438] Der Artikel beginnt mit einem Debian GNU/Linux-Zitat: "To stay free, software must be copyrighted and licensed," und liefert einen kundigen Überblick über die Lizenzmodelle der freien Software. Dann weist er pinkanterweise nach, daß einige zentrale Kritikpunkte an 2B UCC rsp. jetzt UCITA auch dort verwendet werden und auch unerläßlich für freie Software sind. Pikant, da Gomulkiewicz Vorsitzender der damals noch UCC-2B-Arbeitsgruppe der *Business Software Alliance* sowie leitender Unternehmensanwalt von Microsoft ist. Sein zentraler Punkt ist, daß es sich bei freien Lizenzen um "non-negotiated, standard-form, take-it-or-leave-it licenses" handelt, deren Rechtmäßigkeit die UCITA ja gerade absichern soll. Auch am Punkt der Garantiepflicht treffen sich die Interessen von Industrie und Freien. Beide sind gleichermaßen "unwilling to assume the risk of a multi-million dollar class action law suit" und möchten die Möglichkeit von Software-Entwicklern bewahren, "to freely allocate risk." Seine Conclusio zur Zukunft der Open Sourc-Software: "Licensing will be at the center of its success or failure. Article 2B should provide a contract law regime that allows revolutionaries like the open source hackers to succeed."

Das die freie Software der Software-Industrie, gegen deren Schließungsmechanismen sie sich gegründet hat, unfreiwillige Schützenhilfe leistet, ist ein Treppenwitz der Geschichte. Tatsächlich würde die freie Software jeglichen Schutz verlieren, wenn Massenmarktlizenzen invalidiert würde, und im Bankrott enden, wenn sie gesetzlich verpflichtet wäre, eine Garantieleistungsinfrastruktur zu unterhalten. Nach Aussagen von Eben Moglen könnte es im Spätsommer 2000 wegen eines Verstoßes durch ein nichtgenanntes bedeutendes internationales Software-Haus erstmals zu einer gerichtlichen Überprüfung der GPL kommen.[439] Wenn es dabei auch um die Frage der grundsätzlichen Gültigkeit von Massenmarktlizenzen geht, könnte -- nächste Stufe des Treppenwitzes -- die FSF Rückendeckung von Microsoft und den UCITA-Befürwortern bekommen -- falls Microsoft nicht selbst dieses Software-Unternehmen ist.

Schwerwiegende Konsequenzen für die freie Software und ihre Lizenzen sind von der UCITA auf den ersten Blick nicht zu erwarten. Auch sie fallen unter den Schutz der Vertragsfreiheit. Gomulkiewicz nennt die 'Lückenfüller-Regel' des UCITA, die bei Abwesenheit von Vereinbarungen zwischen den Vertragsparteien wirksam werden. Auch das, was die freien Lizenzen nicht explizit festlegen, würde durch Standardregeln eingefüllt. Doch zum einen darf man annehmen, daß die Software-Industrie ihr Bestes dafür gegeben hat, daß diese für Informationsanbieter möglichst harmlos ausgefallen sind, und zum anderen kommen Kriterien wie 'Angemessenheit' und 'redliche Verkehrsübung' zur Geltung, die

---

[437] An anderer Stelle fügt sie hinzu, daß auch die Kunden kommerzieller Software-Unternehmen kein Interesse daran haben könnten, aufgrund von Garantieansprüchen, Prozessen und Haftungsleistungen höhere Preise für die Software zu zahlen. (Kunze 3/1997)

[438] Gomulkiewicz 1999

[439] Powell 6/2000

dem freien Software-Modell -- mit über 15 Jahren Praxis und millionenfacher Verbreitung -- ein schwer zu bestreitendes 'Gewohnheitsrecht' verleihen.

Generell ist jedoch damit zu rechnen, daß durch das UCITA das Klima, in dem wir Wissen schaffen und austauschen, rauer wird. Hat man früher ein Buch gekauft, wenn man die darin enthaltene Information haben wollte, so kauft man heute eine Lizenz -- die handelbare Instantiation von Wissen. "If information ever wanted to be free, it must have changed its mind because under UCC 2B, information seems intent on being licensed."[440] Der schwerwiegenste Aspekt des UCITA ist, daß es den Trend einen gewaltigen Schritt vorantreibt, Vertragsrecht über Urheberrecht dominieren zu lassen (und damit US-Landesrecht über Bundesrecht[441]). Zwar heißt es in Ziff. 105 UCITA, daß Bundesgesetze Vorrang haben vor den UCITA der Länder und natürlich allen mit ihnen kompatiblen Lizenzen, doch ausgerechnet für das Copyright scheint das nicht zu gelten. Das Wort 'Copyright' taucht in dem länglichen Gesetzestext nur viermal peripher auf, obgleich das UCITA an zahlreichen Stellen in den Geltungsbereich des Copyright-Rechts hineinragt. Während z.B. ein Copyright nach Ablauf einer Frist verfällt, d.h. der Eigentumsstatus des Wissens sich verändert, kann nach UCITA ein Lizenzvertrag den Kunden auf alle Zeiten binden. Die Aufgaben der Bibliotheken, wie Zugänglichmachung und Erhaltung von Wissen, können vertraglich und technisch verunmöglicht werden.[442] Was das Coypright nach der *Fair Use*-Doktrin erlaubt, kann eine Lizenz verbieten. Anbieter von Information können ihre Kunden und auch ihre Autoren zwingen, auf Rechte zu verzichten, die sie nach Copyright-Gesetz haben. Wem's nich gefällt, der kann ja wo anders kaufen oder verkaufen. Der Markt wird's schon regeln.

Samuelsons Fazit: bei der Aushebelung des Copyright-Rechts durch Massenmarkt-Lizenzen gehe es darum, "whether copyright owners can have their cake and eat it too." Copyright schreibt mit Verfassungsmandat eine Balance zwischen den Rechten der Autoren und der Öffentlichkeit vor. Diese Balance sei Angriffen durch die überbreite Copyright-Gesetzgebung der jüngsten Zeit, einen Schutz für Datenbanken und nun durch Vertragsrecht ausgesetzt. "We shouldn't let them get away with it. If publishers want the rights that copyright confers, they must take the responsibilities along with the rights."[443] Man kann hinzufügen: Wenn Autoren die Kontrolle über die Nutzung ihrer Werke weitgehend aufgeben, müssen sie von den Verantwortlichkeiten, denen kommerzielle Software-Hersteller unterstehen sollten, freigestellt werden.

---

[440] Samuelson 1998

[441] Wie Samuelson feststellt, haben Landesgerichte wenig Erfahrung mit Coypright, für das die Bundesgerichte zuständig sind, und werden daher geneigt sein, Copyright-rechtliche Aspekte in lizenrechtlichen Verfahren außen vor zu lassen. (Samuelson 1998)

[442] für eine der zahlreichen Kritiken an der UCITA aus der Bibliothekswelt s. Statement of James G. Neal, Dean, University Libraries - Johns Hopkins University, February 3, 2000, http://www.arl.org/info/frn/copy/nealstmt.html

[443] Samuelson 1998

Gesellschaftliche Potentiale freier Software

Freie Software bietet fast allen Nutzern Vorteile. Ihre ökonomischen und technischen Vorzüge übersetzen sich natürlich auch in volkswirtschaftliche Effekte. Zwei aus öffentlicher Sicht bedeutsame Bereiche sollen im Folgenden hervorgehoben werden, ihre Bedeutung in der Bildung und für ärmere Gruppen und Länder. Doch auch der öffentliche Sektor selbst beginnt freie Software zu nutzen. Vielleicht stehen quelloffener Code und der Trend zu einer für den Bürger transparenteren Verwaltung mit weitgehendem Akteneinsichtsrecht ja in morphogenetischer Beziehung zueinander.

Ebenso wie Firmen geben auch Behörden neben praktischen Gründen für den Einsatz freier Software häufig an, daß sie die Abhängigkeit von einzelnen Herstellern vermeiden wollen. “Scandinavia, Germany, and France are some of the main centers of Linux use. Some people say that this is because companies and the government want to avoid becoming too dependent on U.S. -- read Microsoft -- products.”[444]

Ein Beispiel für mögliche Proleme bot sich 1998 in Island, das, um seine Schrift in der digitalen Welt zu erhalten, eine Unterstützung für Isländisch in Microsoft Windows implementiert haben wollte, und sogar bereit war, dafür zu bezahlen. Microsoft sah den Markt als zu klein an und winkte ab. Ohne Zugang zum Quellcode und ohne das Recht, ihn zu modifizieren, ist das Land vollkommen abhängig von Microsofts Gnade.[445] Daraufhin wurde ein Projekt gestartet, die Sprachunterstützung in GNU/Linux zu implementieren, was dank seiner Freiheiten problemlos möglich war, und die öffentliche Verwaltung migrierte auf das freie Betriebssystem. Die Quelloffenheit führt dazu, daß freie Software in einem viel größeren Umfang lokalisiert wird als proprietäre.[446]

Der Staat als Marktregulierer hat die Aufgabe, Monopolauswüchse zu verhindern und kulturelle und Marktvielfalt zu gewährleisten. Immer mehr seiner Organe machen das vor, was sie predigen sollten. In Finnland gibt es bereits Behörden, die komplett mit GNU/Linux arbeiten. In Frankreich soll Quelloffenheit zum Ausschreibungskriterium für Software-Beschaffung werden.[447] Ähnliche Initiativen gibt es in Dänemark und Brasilien. In Deutschland setzten sich u.a. das Bundeswirtschaftministerium, das Bundesamt für Sicherheit in der Informationstechnologie und die Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik für den Einsatz freier Software ein. Der KBSt-Brief aus dem Februar 2000 spricht ein klares Votum dafür aus und gibt praktische Handreichungen.[448] Auf der Konferenz “Effizienter Staat 2000” im April wurde von einer Dienststelle mit 260 Mitarbeitern berichtet, die durch die Umstellung auf Linux gegenüber einer NT-Lösung fast 50.000 DM eingespart hat.[449] Auch der EU-Kommissar für die

---

[444] Kalle Dalheimer, zitiert in: Open Source Software in Europe, http://www.intraware.com/ms/mktg/indaa/itkc/osseurope.html

[445] Walsh 1998, s.a. Vermeer 1998

[446] ein Projekt zur Lokalisierung von GNU/Linux in indischen Sprachen: http://www.freeos.com/indianlinux/ -- KDE Internationalization Page: http://www.kde.org/i18n.html -- das Omega Typesetting System ist eine Modifikation des freien Satzsystems TeX, “designed for printing all of the world's languages, modern or ancient, common or rare”: http://www.serg.cse.unsw.edu.au/DoSE/research.html

[447] Lettice 10/1999. Die beiden Gesetzentwürfe: http://www.senat.fr/grp/rdse/page/forum/texteloi.html; eine öffentliche Debatte dazu im Forum: http://www.senat.fr/Vforum/5/forum.html

[448] KBSt-Brief Nr. 2/2000

[449] s. Gehring 6/2000

Informationsgesellschaft Erkki Liikanen kritisiert die Nichtverfügbarkeit des Sourcecodes kommerzieller Produkte, stellte eine Bevorzugung offener Software bei Ausschreibungen in Aussicht und empfahl die Förderung von Open-Source-Projekten.[450]

Freie Software in der Bildung

Computer-Bildung mit proprietärer Software heißt unweigerlich, daß die Schüler und Studenten lernen, Menü-Punkte zu bedienen. Mit einem 'mächtigen' modularen Betriebssystem wie Unix lernen sie mehr über die Funktionsweisen eines Computers. Durch die Offenheit der Quellen lassen sich diese studieren. Mehr noch erlaubt es die Lizenz, damit zu experimentieren, Änderungen vorzunehmen und dann zu schauen, ob die neu-kompilierte Software das tut, was man erwartet hat. Vor allem autodidaktische Neigungen werden von quelloffener Software angestachelt.

Die Unabhängigkeit von einzelnen Herstellern und die Vermittlung von Computerkenntnissen als Kulturtechnik, nicht als Kompetenz, die Software dieser Hersteller bedienen zu können, sieht auch Peter Bingel als seinen Bildungsauftrag. "Freie Software schafft, so denke ich, einen informationellen Freiraum. Entscheidend ist, dass sie für jeden verfügbar und zu besitzen ist und mit ihrer Hilfe ein von kommerziellen Interessen unabhängiger Raum entsteht. Und ich denke, dieser freie Raum ist besonders für den Bildungsbereich wichtig. [...] Alleine durch Freie Software ist man in der Lage, sich informationstechnologisch von der Vorherrschaft einiger weniger Global Players zu lösen. Dies ist eine Botschaft, die wir als Lehrer unseren Schülern unbedingt mitgeben sollten."[451] Bingel warnt insbesondere vor der Kommerzialisierung der Wissensbasis durch das Engagement von Firmen wie Microsoft, Apple, IBM und Sun im Bereich von Lern-Software und bei der Errichtung kommerzieller Bildungsangebote. Er schätzte Mitte 1999, daß deutlich über 10% aller deutschen Schulen Linux einsetzen. Meist geht das auf die Inititive einzelner Lehrer zurück, die aber im *Verein Freie Software und Bildung*[452] Unterstützung finden. In der Regel wird Linux als Server eingesetzt, aber Bingel betont, wieviel Unterricht-geeignete Software es bereits gibt.[453] Auch er spricht den Vorteil für Schüler an, deren Muttersprache nicht Deutsch ist, KDE in verschiedenen Sprachen laufen zu lassen. Nur freie Software erlaubt es außerdem, den Schülern Kopien der Programme mit nach Hause zu geben.

Emre Demiralp berichtet von einem Beispiel aus der höheren Bildung in der Türkei. Im Fachgebiet Elektrotechnik der technischen Universität Istanbul wurde 1998 eine Version des türkischen Linux (Turkuvaz) entwickelt. Bis 1991 standen den Studenten elf PCs zur Verfügung. Es gab dauernd Probleme mit Viren und Systemzusammenbrüchen. Ab 1992 stand ein SUN-Rechner zu Verfügung. Über das BITNET wurde die Leitung des Fachbereichs auf ein neues Betriebssystem namens Linux aufmerksam. Die Software war kostenlos und es gab genug Informationen im Internet. Die Angehörigen der Universität stellten jedoch fest, daß die Administration der Rechner keine leichte Aufgabe war. Es war zeitintensiv und fast unmöglich, mit wenigen Administratoren hunderte von Studenten zu

---

[450] s. z.B. seine Keynote Speech auf der European Approach Information Security Solutions Europe, Berlin, 4 October 1999, http://europa.eu.int/comm/dg03/speeches/liikanen/EL041099.html

[451] Peter Bingel, Wizards 7/1999

[452] http://fsub.schule.de

[453] s. http://fsub.schule.de/oss/Software/software.html

betreuen. Da es zu teuer war, Fachkräfte für die Systempflege einzustellen, wurden Studenten als Administratoren gewonnen. Sie sollten die Systeme warten und konnten zugleich mehr lernen. Heute sind schätzungsweise 100 Studenten an der Pflege des Systems beteiligt. Sie opfern ihre Freizeit, um ihr Wissen zu erweitern, wodurch sie bei ihrem Abschluß eine zusätzliche Auszeichnung erhalten können. Außerdem werden Informationen über LINUX und verwandte Themen gesammelt und Studenten zur Verfügung gestellt. Alle zwei Monate wird auch ein türkisches Online-Magazin publiziert. Mittlerweile bestehen die Rechnerpools aus 70 Pentium 166ern, von denen 50 unter Linux laufen. Die Wartung wird durch die Studierenden vorgenommen. Das System selbst wird von ca. 500 Studenten 24 Stunden genutzt, die längste Zeit ohne einen einzigen Crash war 90 Tage.[454] Ähnliche Gruppen und Initiativen wie den Verein Freie Software und Bildung gibt es auch in anderen Ländern.[455]

Freie Software in den Nicht-G7-Ländern

Eine weitere besonders betroffene Nutzergruppe sind die weniger Betuchten dieser Welt. Der offenkundigste Vorteil für sie ist, daß Software-Kosten gespart werden können. Aber auch bei der Hardware ist es nicht erforderlich, das Wettrennen um die allerneueste Technologie mitzumachen, da GNU/Linux und andere freie Software in Versionen entwickelt wird, die ältere Hardware einsetzbar macht. Ein Intel-386er, Eine Rechnergeneration, die die meisten kommerziellen Nutzer längst ausgemustert haben, genügt den meisten heutigen Anforderungen. Das erlaubt es Organisationen wie der *Computer Bank* Hardwarespenden von Firmen, Behörden und Privatleuten entgegenzunehmen, sie mit freier Software auszustatten und an diejenigen umzuverteilen, die sich selbst solche Rechner nicht leisten können. *Computer Bank* ist ein australisches Projekt, das alte Niedrigverdienende, Community-Gruppen und benachteiligte Schulen mit alten Rechner ausstattet.[456]

Ende 1998 gab die mexikanische Regierung Pläne für ein *Scholar Net Program* bekannt, das in den folgende fünf Jahren Linux an 140.000 Schulen im ganzen Land mit fast 20 Millionen Schülern einführen wird. Die mexikanischen Schüler sollen Web- und eMail-Zugang erhalten, mit Textverarbeitung und Tabellenkalkulation vertraut gemacht werden und im Computer-Raum ihre Recherchen und Hausaufgaben machen können -- umso wichtiger, in einem Land, in dem nur eine verschwindende Minderheit einen Computer zu Hause hat. Parallel dazu bietet ein Projekt für eine digitale Bibliothek den Schülern die Möglichkeit, auf Lehrbücher im Netz zuzugreifen. Die Wahl von Linux begründete der Projektleiter Arturo Espinosa Aldama damit, daß die Kosten für kommerzielle Software nicht tragbar gewesen wären. Eine Microsoft-Lösung hätte das Land 124 Millionen Dollar gekostet. Nicht allein das Budget sprach für die Einführung von LINUX, sondern auch die bessere Verläßlichkeit, Anpassungsfähigkeit und Effizienz im Vergleich zu kommerzieller

---

[454] Demiralp 1998

[455] s. z.B. OFSET, the Organization for Free Software in Education and Teaching, http://www.ofset.org -- den *Linux in education report*, http://www.seul.org/edu/reports.html -- France to install Linux in schools, Okt. 1998, http://linuxtoday.com/stories/515.html

[456] http://www.computerbank.org.au/ mit Links auf ähnliche Projekte in Canada, Frankreich und den USA

Betriebssystem-Software.[457] Scholar Net schließt Fernunterricht über Fernsehen (RED EDUSAT) und eMail ein. Schulen und Schüler erhalten eMail-Adressen und persönliche Webseiten. Koordiniert wird das Projekt von der *Universidad Nacional Autónoma de México* (UNAM) und dem *Instituto Latinoamericano de la Comunicación Educativa* (ILCE). Das Geld stammt von der Regierung und aus Spenden. In der ersten Phase hat das Projekt eine GNU/Linux-Distribution (mit Servern, wie sendmail, httpd, diald und squid, und für die Workstations GNOME-Applikationen, wie gwp, gnumeric und Netscape) namens Red Escolar[458] zusammengestellt und sie bei den Bildungsbehörden und Schulen in den einzelnen Bundesländern bekanntgemacht. Möchte eine Schule von dem Angebot Gebrauch machen erhält sie in der nächsten Phase Unterstützung bei der Installation auf der vorhandenen Hardware und Training. Espinosa ist über die Rückendeckung durch die Behörden erfreut: "We thought there would be some resistance from the big bosses but, due to the attention GNU/Linux is being paid in the mainstream computing media, the University is now considering GNU/Linux as a valid option and we are being accepted because we do have something to show."[459] Im Bildungsministerium denkt man bereits über die nächste Phase nach, in der die Entwicklung von Bildungs-Software im Vordergrund stehen soll.

Während in Mexico und Brasilien die Regierung diesen Kurs fördert, geht die ersten Schritte in die freie Software meist auf *Grassroots*-Initiativen zurück, wie ein Beispiel aus Malaysien zeigt. Die Vereinigung der Thalassaemie-Betroffenen war mit der Bitte an die Malaysische *Open Source Group* herangetreten, ihnen bei der Errichtung einer eCommunity zu helfen. Geld aus einem staatlichen Förderprogramm war in Aussicht gestellt, doch Papier ist geduldig. Statt dessen fand sich ein 155er Pentium mit 500 MB Festplatte -- nach allen Standards auch das schon wenig mehr als Computer-Schrott. Darauf installierte die Gruppe das schlanke FreeBSD, dazu Apache, Majordomo, Sendmail, Bind und Perl. Damit hatte die Community eine Web-Site, Mailinglisten, ein Web-basiertes Diskussionsforum und ein Formular, um sich für Veranstaltungen zu registrieren.[460] Sobald das Geld eintrifft, werden sie die Hardware und das Informationsangebot erweitern. Die Open Source Group wird der *Community* dabei helfen, ihnen beibringen, das System selbst zu unterhalten und dann weiterziehen zu einer anderen Non-Profit-Community.

Afrika hat die geringste Durchdringung mit Computern und Internet. Während 1995 nur acht afrikanische Staaten über einen Internet-Zugang verfügten, waren es Ende 1997 bereits zweiundvierzig. Seit 1999 gibt es in allen Ländern Afrikas, außer in Somalia, Internetzugang.[461] Die Bandbreite wird in den nächsten Jahren u.a. mit Hilfe von neuen Satelliten und eine Unterwasser-Glasfaserkabels, das den gesamten Kontinent umspannt, erheblich erweitert. Nazir Peroz, Sprecher der Fachgruppe 'Informatik und Dritte Welt' der Gesellschaft für Informatik und Dozent an der TU Berlin, berichtete daß Linux an Universitäten in Simbabwe und bei der WHO verwendet wird, da es sich um ein robustes Betriebssystem handelt. Die Lage in Äthiopien ist ähnlich. Peroz' Ansprechpartner in

---

[457] Gabriel Moreno, Linux, el nido nacional del cómputo, Publi.com-News, 17. Nov. 1998, http://www.publi.com/news/1998/1117/x06.htm

[458] http://redesc.linux.org.mx/

[459] Red Escolar FAQ #6, http://redesc.linux.org.mx/en/faq.html

[460] s. http://tam.org.my

[461] Information & Communication Technologies (ICTs) Telecommunications, Internet and Computer Infrastructure in Africa gab im August 2000 an, daß 53 der 54 afrikanischen Länder und Territorien in ihren Hauptstädte Internet-Zugang haben, http://www3.sn.apc.org/africa/

Mosambik sind zwar an freier Software interessiert, scheitern jedoch an grundlegende Problemen der Informatikausbildung und der Ausstattung der Universität. Neben den technischen Möglichkeitsbedingungen sind Wissen und Bildung Voraussetzungen für eine Teilhabe an der globalen Informationsgesellschaft. Die Informatikausbildung an afrikanischen Hochschulen sei zu wenig praxisorientiert. Hier sieht Peroz Herausforderungen für den Informationstechnologie-Transfer. In der *Arbeitsgruppe Computer Information Transfer* unterstützt er afrikanische Dozenten und Studierende über das Internet bei Informatikfragen. So arbeiten die AG-CIT z.B. mit dem Fachbereich Informatik der Universität Simbabwe in Harare zusammen.[462]

Einen gewaltigen Zulauf könnte freie Software in China erhalten. Mit einer Zuwachsrate von zehn Millionen verkauften PCs pro Jahr nimmt die Computerisierung dort rasch zu. Da eine Kopie von Microsoft Windows 98 zu einem Preis verkauft wird, der einem halben Jahreslohn eines Arbeiters entspricht, handelt es sich bei der verwendeten Software überwiegend um illegale Kopien. In diesem Segment herrschten bislang Microsoft-Betriebssysteme vor, obgleich sie der Handhabung chinesischer Schriftzeichen nicht besonders entgegenkommen. Am Software-Institut der Chinesischen Akademie der Wissenschaften wurde im Juni 2000 die erste chinesischsprachige 64-Bit-Version von Linux vorgestellt. Während bei Windows vier bis sechs Tasteneingaben nötig sind, um ein Zeichen aufzurufen, benötigt das neue *Chinese 2000* der Wissenschaftsakademie nur durchschnittlich 2,5 Tasteneingaben. Federal Software, der größte Software-Vertreiber des Landes, verkaufte 1999 sein chinesisches Linux beinah 200.000 Mal, etwa 200 Mal häufiger als das chinesische Windows. Federal bietet denjenigen Bluepoint-Linux zum Preis von 10 Yuan ($2) an, die eine Kopie illegaler Software aushändigen. Verschiedene wichtige Regierungsbehörden beschlossen, das einheimische Red Flag Linux einzusetzen. Auch Red Flag wurde an der Wissenschaftsakademie in Zusammenarbeit mit Compaq entwickelt.[463]

Ein Gutteil der primären Ressourcen der Welt liegen in den ärmeren Ländern des ‘Südens,’ wenngleich auch ihre Ausbeutung häufig in den Händen externer Unternehmen liegt. Das Eigentum an Wissen (wie das an Kapital) dagegen konzentriert sich im ‘Norden’. Das traditionelle Wissen des Südens, z.B. über Pflanzenheilkunde, wird auch noch, ohne Kompensation, enteignet, um es in die Informationsverarbeitungsmaschinerie des Nordens zu speisen. Wo in ärmeren Ländern Computer verfügbar sind, ist häufig auch ‘kostenfreie’ proprietäre Software zu haben, doch die kann man nur unter der Drohung der Einschüchterung und Verfolgung durch Unternehmen und Polizei verwenden. Vor allem die WIPO und die WTO stehen für eine aggressive Agenda der Durchsetzung von geisten Eigentumsrechten in aller Welt. Freie Software bietet auch in dieser Hinsicht eine bessere Alternative.

SatelLife[464] ist eine international *not-for-profit* Organization, die die Kommunikation im Gesundheitswesen in den Entwicklungsländern verbessern hilft. Sie hat dazu ein umfassendes Netz aus Funk, Telefon und dem *Low Earth Orbit* (LEO)-Satelliten HealthSat-2. HealthNet ist ein (*store-and-forward*) FidoNet. Die Software für seine Satelliten-Gateways und die Bodenstationen hat SatelLife auf der Basis von Linux entwickelt.

---

[462] Nazir Peroz, Wizards 7/1999

[463] Dwyer 2000

[464] http://www.healthnet.org/

> “For starters, the staff of Satellife had to seek out and master technologies cheap enough for users in the world's poorest countries but reliable enough to deliver vital medical information fast. And the organization didn't have the funds that corporate IT departments have for equipment and software -- so it used free and open-source software to link users to forums. And as the Internet became a more vital tool, Satellife had to make sure that users without browsers could still get information via the Web. It also used second-hand gear where possible and relied on research institutes and discussion groups, rather than high-priced consultants, for advice.”[465]

Nicht nur NGOs, sondern auch die ‘große’ Entwicklungspolitik hat sich dem *digital Divide*[466] angenommen. Das *Third World Network of Scientific Organizations*, aber auch Organisationen wie die UNESCO, die Weltbank (InfoDev) und USAID betreiben Initiativen, um die IT-Infrastruktur, Ausbildung und Forschung in Entwicklungsländern zu verbessern und Kooperationen zu fördern. Das *United Nations Developement Programme* betreibt seit 1992 das *Sustainable Development Networking Program* (SDNP) eine Initiative um die lokale wie Internet-Netzwerkinfrastruktur in Entwicklungsländern zu fördern und Menschen zu helfen Wissen für eine nachhaltige Entwicklung miteinander zu teilen. “Information and Communication Technologies [ICTs] are now fundamental to dealing with all development issues in developing countries and cuts across UNDPs main areas of concentration. It is a core tool needed to achieve Sustainable Human Development (SHD) and one that can facilitate the 'leap-frogging' of developing countries into 21st century ICTs and SHD goals.”[467] Corel[468] und Red Hat statten das Programm mit ihren Linux-Distributionen aus. Auch die UNESCO verbreitet kostenlose Linux CD-ROMs an wissenschaftliche, Bildungs- und Community-Projekte in Lateinamerika:

> “We believe LINUX can play a very important role in Latin American and Caribbean modernisation, constructing networks to permit a great number of universities, colleges, schools and educational centers, to connect to Internet in order to use this fabulous tool to improve their scientific and cultural levels. In a few words, LINUX is the tool which permits to reduce the ‘technological gap’ between the countries. LINUX permits the acces to ‘the informatics [of] the most advanced’ implemented according to the reduced economic capacities in our region. LINUX is a new way to make informatics, where the most important thing is ‘the technical quality and people solidarity.’”[469]

Auf dem *Colloque Inforoute et Technologies de l'Information* im Oktober 1997 in Hanoi ebenso wie auf der von der UNESCO in Zusammenarbeit mit dem International Council for Science organisierten *World conference on Science* im Juni 1999 in Budapest spielte die Bedeutung von freier Software für die Teilhabe am technologischen Fortschritt eine wichtige Rolle.

---

[465] http://www.data.com/issue/981021/people.html

[466] http://www.DigitalDivideNetwork.org/

[467] http://www.sdnp.undp.org/home.html -- ein guter Nachrichtenüberblick zu Open Source nicht nur in Entwicklungsländern: http://www.sdnp.undp.org/perl/news/articles.pl?do=browse&categories=10

[468] http://www.corel.co.za/pressroom/august_10b.html

[469] http://www.unesco.org/events/latin/cd_linux_ing.html

Wirtschaftliche Potentiale freier Software

> "The Linux community, a temporary, self-managed gathering of diverse individuals engaged in a common task, is a model for a new kind of business organization that could form the basis for a new kind of economy."
> (Harvard Business Review, September 1998)

Die Bewegung der freien Software mag auf den ersten Blick als eine Aberration vom allgemeinen Marktgeschehen erscheinen. Auf den zweiten wird man sie für einen 'Rückfall' in den ursprünglichen Zustand der Software-Branche vor Beginn ihrer Kommodifizierung halten können. Noch genaueres Hinsehen fördert dann jedoch eine Reihe Korrespondenzen und Strukturähnlichkeiten zu Trends der 'offiziellen' Ökonomie zutage.

Seit den sechziger Jahren wird hier ein Strukturwandel von der Industrie- und Warengesellschaft zur Informations- und Dienstleistungsgesellschaft attestiert. Information wird anstelle von Materie zur zentralen industriellen Ressource und Ware. Als einen Umschlagpunkt nannte der Club of Rome in seiner Studie "Grenzen des Wachstums" das Jahr 1970. Die Indikatoren für stofflichen Reichtum, die bis dahin mit dem Bruttoinlandsprodukt korrelierten, entwickeln sich seither umgekehrt proportional: je reicher ein Land, desto geringer der stoffliche Anteil an diesem Reichtum. Im Maße das Internet zur Infrastruktur der Wirtschaft wird, werden Grenzen zwischen Unternehmen und die zwischen Nationalökonomien fließend. 'Virutelle Unternehmen' bestehen aus nicht mehr als einem Planungskern, der für die Dauer eines Projekts Kooperationen mit anderen eingeht und Leistungen nach Bedarf von außen zukauft.

Jeremy Rifkin sagt in seinem neuen Buch das Verschwinden des Eigentums voraus, dessen Besitz durch den Zugang ("Access") zu vor allem geistigem Eigentum ersetzt werde.

> "Unternehmen sind in diesem Übergang vom Besitz zum Zugang schon ein Stück vorangekommen. In einem gnadenlosen Wettbewerb verkaufen sie ihren Grundbesitz, verschlanken ihr Inventar, leasen ihre Ausstattung und lagern ihre Aktivitäten aus; sie wollen sich von jeglichem immobilen Besitz befreien. Dinge, und zwar möglichst viele, zu besitzen wird in der an Schnelligkeit und Flexibilität orientierten Wirtschaft des neuen Jahrhunderts als überholt und lästig betrachtet. In der heutigen Geschäftswelt wird fast alles geliehen, was ein Unternehmen zu seinem Betrieb braucht."[470]

Den gleichen Trend sieht er bei den Verbrauchern.

> "Zwar werden niedrigpreisige haltbare Dinge auch weiterhin gekauft und verkauft werden, teurere Objekte jedoch, Geräte, Autos oder Häuser, werden zunehmend von Anbietern gehalten werden, die den Konsumenten über zeitlich befristete Leasing- oder Mietverträge, Mitgliedschaften und andere Dienstangebote Zugang und Nutzung gewähren."[471]

---

[470] Rifkin 2000
[471] ebd.

Und was wird aus der materielle Produktion, der Landwirtschaft und den nicht-informationellen Dienstleistungen? Hier werde die menschliche Arbeitskraft zunehmend von intelligenten Maschinen ersetzt. 2050, so prophezeit Rifkin, werden nicht mehr als fünf Prozent der erwachsenen Bevölkerung benötigt, um die herkömmlichen Betriebe in Gang zu halten. Die übrigen 95 Prozent (falls es eine Vollbeschäftigung geben sollte) werden dann, so suggeriert er, in der Informationsökonomie arbeiten, vor allem in der Kulturindustrie, die die letzte Stufe des Kapitalismus darstelle.

Wie die Arbeitsbedingungen im Herzen der High-Tech-Industrie, im Silicon Valley, heute aussehen, umreißt Netzaktivist Florian Schneider so:

> "'Nettokratie' ist eine der jüngsten Wortschöpfungen, die den Blick auf die soziale Zusammensetzung der Informationsgesellschaft lenken soll. Den von Arthur Kroker schon 1994 zur 'virtuellen Klasse' erhobenen Entrepreneurs steht ein Heer von Netzsklaven gegenüber, die sich bei den Start-Ups in den Silicon-Somethings verdingen [...] Die Arbeitskräfte, sagt Andrew Ross, der als Direktor des American Studies Program an der New York University[472] die Situation im Silicon Alley untersucht hat, seien zur Hälfte Werkvertragsarbeiter, die vor allem darauf angewiesen seien, daß ihre Aktienanteile steigen. Das Durchschnittseinkommen liege mit 50.000 US Dollar ungefähr bei der Hälfte dessen, was in den alten Medien verdient werde. Bemerkenswert ist, daß ausgerechnet Künstler mit ihrem flexiblen und selbstlosen Arbeitsethos das Rollenmodell für die 'freiwillige Niedriglohn-Armee' abgeben. Der 'Glamour der Boheme' kommt nach Ross einer Einladung zur Unterbezahlung gleich. [...]
>
> Daß die 'New Economy' aber nicht nur hochqualifizierte Jobs hervorbringt, sondern vor allem Unmengen von vergleichsweise banalen Tätigkeiten wie Telefonieren, Pizza-Bringen oder Saubermachen schafft, wird in den gegenwärtigen Debatten gewöhnlich unterschlagen. In den USA machen seit einigen Wochen Tausende von Reinigungskräften mit einer Streikwelle bisher ungekannten Ausmaßes auf sich aufmerksam. Die 'Janitors', die die Office-Türme jede Nacht von den Überresten der immateriellen Arbeit reinigen, sind meist lateinamerikanische Einwanderer. Die 'Janitors' kämpfen um eine sukzessive Anhebung ihrer Hungerlöhne, für die die boomenden High-Tech-Firmen allerdings keine Verantwortung übernehmen wollen. Für die schmutzigen Geschäfte sind nämlich Subunternehmer zuständig."[473]

Auf der Konsumentenseite sieht der führende deutsche Mikroökonom Norbert Szyperski durch Online-Auktionen den Markt sich in einen 'Basar' im eigentlichen Sinne verwandeln: die Güter haben keine feste Ausschilderung, alle Preise werden ausgehandelt. Mit dem neuen Ort des Marktes entstehen neue Regeln, neue Verhältnisse zwischen Anbieter, Käufer und Ware und neue Fragen: "Wie kann man Übertragung sicherer machen? ... Wie kann man ein geschäftliches Vertrauensverhältnis in dieser Medienwelt etablieren, wie man das

---

[472] http://www.nyu.edu/gsas/program/amerstu/corefac.html

[473] Florian Schneider, Subject: [rohrpost] tulpenwahn, auf rohrpost@mikrolisten.de, 07 Jun 2000 20:23:37 +0200

üblicherweise mit seinen Stammlieferanten oder Stammkunden haben konnte? ... Wie macht man internationale Kleingeschäfte? Welche Rechte gelten da?"[474]

Verschiebt die informationstechnologiegestützten Ökonomie den Schwerpunkt von materiellen zu immateriellen Gütern, so gleichzeitig den vom Verkauf von Werkstücken zur Lizenzierung von Nutzungen. "Heute schon gibt das reiche obere Fünftel der Weltbevölkerung für den Zugang zu kulturellen Erlebnissen genausoviel aus wie für Fertigerzeugnisse und Dienstleistungen."[475] Mit diesem Wandel geht auch der vom Produkt zum Prozeß einher. Bislang wurde eine Software als Werkzeug angesehen, das -- zumindest bis zum nächsten *Upgrade* -- statisch bleibt. Nach dem neuen Verständnis steht auch im kommerziellen Teil der Branche die Implementierung, die laufende Anpassung und die Mitarbeiterschulung im Vordergrund. Software könnte statt als Produkt als eine Dienstleistung zur Generierung, Distribution, Manipulation und Archivierung von Informationsflüssen verstanden werden. Perry Barlow schrieb 1994 in seinem klassisch gewordenen Aufsatz "The Economy of Ideas", Information sei eine Aktivität, eine Lebensform und eine Beziehung.[476] Die Aussage ist der freien Software-Bewegung wie auf den Leib geschneidert.

Wird also die Linux-Commmunity, wie die Harvard Business Review schrieb, zum Modell einer neuen Art von Ökonomie? Ist sie die Avantgarde eines generellen Stukturwandels oder nur ihr Nutznießer? Oder ist sie das Trüffelschwein, das im Möglichkeitsraum des Internet stöbert und dabei Schätze hervorholt, aus denen sich die kapitalistische Maschinerie nimmt, was sie zu ihrer Verjüngung braucht? Oder ist im Grunde das, was die freie Software-Welt macht, dasselbe, was seit den achtziger Jahren auch in den Unternehmen geschieht?

Eine allgemeine Tendenz zu Dezentralisierung, Abbau von Hierarchien und offenen Systemgrenzen ist auf jeden Fall nicht zu übersehen. Globalisierung und Flexibilisierung der Wirtschaftsstrukturen sind vielfach verzeichnete Trends. Die Projekte der freien Software übetreffen jede management- und organsiationstheoretische Vision an Ortsverteiltheit, lockerer Kooperation und zwangloser Koordination. Genauso wie die Frage nach dem 'Standort' eines multinational operierenden Unternehmens wenig Sinn macht, kann man auch von den Software-Projekten nicht sagen, sie seien amerikanisch, deutsch oder sonstwie nationalstaatlich zuzuordnen. Ihr 'Standort' ist das Internet.

Auch das Konzept vertikaler Kooperation ist der Wirtschaft nicht fremd. Der 'Wertschöpfungspartner auf der Nachfrageseite', wie es im Ökonomenjargon heißt, also der Anwender, wird zum 'Koproduzenten'.

> "die freie Mitwirkung ist etwas, was praktisch wie eine Epidemie durch die gesamte dienstleistende und wissensintensive Industrie hindurchgeht. Nicht umsonst versucht man Kooperation und *Competition*, also Wettbewerb, auch begrifflich in *Cooptition* zu fassen, weil wir nicht mehr so scharf sagen können: Wer ist eigentlich mein Gegner oder mit wem mache ich gemeinsame Entwicklung und mit wem treffe ich mich nur beim Vertreter, möglicherweise beim Kunden, als Konkurrent? Koproduktion ist der Begriff für die Dienstleistung schlechthin. Wenn Sie heute z.B. das Gesundheitswesen neu diskutieren, sprechen Sie nicht mehr über die Frage: Was

[474] Prof. Norbert Szyperski, Wizards 7/1999

[475] Rifkin 2000

[476] Barlow 1994

> sollte der Arzt mit dem Patienten tun, wenn der in die Klinik oder in die Sprechstunde kommt? Sondern wir gehn davon aus, daß derjenige, der noch leben will, selber Koproduzent seiner Gesundheit sein muß."[477]

Software-Anwender, die ein Programm benutzen, testen, Fehler zurückmelden und Verbesserungsvorschläge machen, sind Koproduzenten, ob von Microsoft-Windows oder von GNU/Linux. Wie oben gezeigt,[478] ist nicht einmal die Kooperationsgemeinschaft oder, wenn man so will, Allmendgenossenschaft ein Privileg der freien Software. Auch herkömmliche Unternehmen errichten und pflegen ihre *gated Communities*. Bleibt also der Preis?

> "Traditionell bestimmt sich der Preis einer Ware aus den Faktoren Angebot und Nachfrage. Da die Freie Software per Definition beliebig kopiert und verteilt werden darf, und weil die Reproduktion im Zeitalter von CD-ROM und Internet mit keinen nennenswerken Kosten verbunden ist, wird das Angebot beliebig groß. Der Preis für die Software muß demnach selbst bei überwältigender Nachfrage beliebig klein werden. In der Realität befindet er sich tatsächlich auf dem Niveau des Selbstkostenpreisen für die Reproduktion. Einen Internetanschluß vorausgesetzt ist Freie Software ohne zusätzliche Kosten erhältlich."[479]

Die nahezu kostenlose Distributionsmöglichkeit gilt auch für proprietäre Software, doch die ist eben in der Regel nicht kostenlos. Es ist gerade der *Free Beer*-Aspekt der freien Software, der gestandenen Ökonomen Rätsel aufgibt. Meist ordnen sie das Phänomen zunächst dem Marketing zu. Freie Software entspräche demnach einem Werbegeschenk, das Kunden für andere Produkte oder Dienstleistungen gewinnen soll, etwa einem Mobiltelefon, das in der Erwartung kostenlos abgegeben wird, daß die Grund- und Verbindungsgebühren über den Vertragszeitraum hinweg nicht nur die Kosten des Gerätes, sondern einen Profit darüber hinaus einspielen.[480] Auch Rifkin sieht dies als einen generellen Trend. "Im klassischen Industriezeitalter wollten Unternehmen vorrangig ihre Produkte verkaufen; kostenlose Servicegarantien setzten Kaufanreize. Heute ist dies geradezu umgekehrt. Immer häufiger geben Unternehmen ihre Produkte buchstäblich umsonst ab: Sie hoffen statt dessen auf langfristige Servicebeziehungen zu ihren Kunden."[481]

Solche Mechanismen findet man an zahlreichen Stellen in der emergierenden Internet-Ökonomie. Kostenlos zugängliche Information in Form von redaktionell erstellten Magazinen, aufbereiteten Portalen und thematischen Diskussionen generiert eine Aufmerksamkeit, die in Form von Bannern, Listen potentieller Kunden und anderen Mechanismen an die Werbeindustrie verkauft werden kann. Information wird zur Markteinführung und zum Betatesten eine Zeit lang kostenlos angeboten, um dann Gebühren dafür zu erheben. Proprietäre Software wird als Demoversion mit eingeschränkter Funktionalität oder Laufzeit verschenkt, um zum Kauf der Vollversion zu locken. Client-Software (Web-Browser, Viewer und Player für Text- (z.B. Adobe Acrobat), Audio- und Video-Formate (z.B. Real-Player)) wird kostenlos abgegeben, um Informationsanbieter zu

---

[477] Prof. Norbert Szyperski, Wizards 7/1999

[478] unter "Unfreie Lizenzen"

[479] Hetze 1999

[480] so z.B. Szyperski, Wizards 7/1999

[481] Rifkin 2000

bewegen, die Editier- und Server-Software für diese Formate zu erwerben. Klassische Werbegeschenke, wie kleine Werkzeuge oder Spiele, sollen Kunden auf Webseiten locken und helfen, einen Firmen- oder Markennamen zu etablieren.

Dabei handelt es sich natürlich nicht um freie Software. Weder ist hier der Quellcode verfügbar, noch darf er modifiziert werden. Doch mancher Nutzerin mag es gleich sein, ob ihre Programme von Microsoft oder von der FSF kommen, solange sie nur kostenlos sind. Auch freie Software wird in diesem Marketing-Sinne als *Add-On* zur Aufwertung von proprietären Produkten verwendet. So wird beispielsweise der Apache Webserver von IBM zusammen mit dessen Server-Systemen vertrieben. Für SCO-Unix gibt es die *Skunkware-CD*, auf der freie Software für dieses Betriebssystem verteilt wird. Computerfachbüchern und Zeitschriften enthalten als *Added-Value* CDs mit freier Software.

Ein anderes ‘Geschenkmodell’ benutzen Unternehmen, die bestehende oder neue Software *open-sourcen*.[482] Gründe dafür können sein, daß sie ein neues Produkt in einem saturierten Marktsegment einführen und dazu Aufmerksamkeit generieren, daß sie Entwicklerressourcen aus der freien Szene einbinden wollen oder daß ihr Business-Plan sich nicht auf den Verkauf von Software, sondern von Dienstleistungen stützt. Es handelt sich dabei also um traditionelle Firmen, deren Aktivitäten in den Bereich der freien Software hineinragen, während umgekehrt die freie Software-Szene sich geldökonomische Tätigkeitsfelder erschließt.

Werbegeschenke haben, im Gegensatz zu reiner Werbung einen Gebrauchswert. Sie werden kostenlos abgegeben und auf anderen Wegen, letztlich vom Käufer der beworbenen Ware bezahlt. Die freie Software wäre in Analogie das Geschenk, das die Dienstleistungen (Distribution, Support usw.) bewirbt, deren Möglichkeitsbedingung sie zugleich ist. Die Analogie findet darin ihre Grenzen, daß Werbegeschenke in einer engen geldökonomischen Schleife von ‘Geschenk’ und erhofftem Ertrag desjeniegen stehen, der das Geschenk vorfinanziert. Freie Software lebt jedoch weiterhin wesentlich davon, daß viele Menschen Arbeit in sie stecken, die die Reproduktion ihrer Arbeitskraft anderweitig sichern können.

Bei allen Ähnlichkeiten bleibt der zentrale Unterschied: freie Software wird nicht aus pekuniären Gründen erstellt. Perry Barlow schrieb 1994 unter der Zwischenüberschrift “Information ist ihre eigene Belohnung”:

> “And then there are the inexplicable pleasures of information itself, the joys of learning, knowing, and teaching; the strange good feeling of information coming into and out of oneself. Playing with ideas is a recreation which people are willing to pay a lot for, given the market for books and elective seminars. We'd likely spend even more money for such pleasures if we didn't have so many opportunities to pay for ideas with other ideas. This explains much of the collective ‘volunteer’ work which fills the archives, newsgroups, and databases of the Internet. Its denizens are not working for ‘nothing,’ as is widely believed. Rather they are getting paid in something besides money. It is an economy which consists almost entirely of information.
>
> This may become the dominant form of human trade, and if we persist in modeling economics on a strictly monetary basis, we may be gravely misled.”[483]

---

[482] s.o. unter “Unfreie Lizenzen”

[483] Barlow 1994

Daß Menschen Dienstleistungen lieber mit Gegenleistungen als mit Geld bezahlen, motiviert auch die Tauschringe, die in vielen deutschen Städten und andernorts sprießen. Sie sind dem Phänomen der freien Software noch am ehesten verwandt in der gemeinsamen Überzeugung, daß es neben dem Geld noch andere würdige Werte gibt. "[N]othing kills the notion of community faster than putting a price on it."[484] Auch bei Oekonux, einer Gruppe, die über die Ökonomie von GNU/Linux diskutiert, herrscht die einhellige Auffassung, daß sich Geld und freie Entfaltung gegenseitig ausschließen.

> "Der Aspekt des 'nicht-kommerziellen' [...] oder der Entwicklung außerhalb von Verwertungszusammenhängen wie ich das nennen würde, ist IMHO *der* entscheidende Aspekt. [...] Wenn 'free beer' für Verwertungsfreiheit steht, dann bin ich auch für free beer, so hats Stallman allerdings nicht gemeint.
>
> Warum verwertungsfrei? Nur außerhalb solcher Verwertungszusammenhänge (sprich: außerhalb von Lohnarbeit) ist wirkliche Entfaltung des kreativen Menschen möglich. Das hat auch Eric Raymond erkannt, irgendwo auf seiner Seite zitiert er Untersuchungen, aus denen hervorgeht, das Menschen viel unproduktiver sind, wenn sie 'für Geld arbeiten' als wenn sie 'for fun' sich entfalten. Verwertung und Entfaltung ist ein unaufhebbarer Widerspruch![485]

Nachdem das gesagt ist, bleibt die Frage, womit wird denn nun in der freien Software Geld verdient? Eine frühe Analyse der Nutzungsschritte frei weiterverbreitbarer Software und der jeweiligen Möglichkeit von Einkünften findet sich bei Peter Deutsch.[486] Die wichtigsten Chancen für Software-Arbeiter sieht er in der Unterstützung von Anwendern bei Installation und Bedienung sowie bei Anpassung und Erweiterung. Da der Quellcode offen ist, bieten sich dazu auch für Dritte Gelegenheiten, die bei proprietärer Software auf den Hersteller und lizenzierte Partner beschränkt sind. Deutsch weist auch auf die unter Gewinngesichtspunkten widersprüchliche Position der Autoren hin. Es sei in ihrem Interesse, rasch unfertige Software mit mangelhafter Dokumentation zu verbreiten, da dadurch ihre Chancen für Dienstleistungen steigen. Ist die Software von guter stabiler Qualität und gut dokumentiert, sei der Bedarf nach Support minimal, besonders, da sich freie Software meist an Entwickler richte. Reine Anwender-Software gebe es nur in sehr geringem Umfang. Tatsächlich sei aus diesen Gründen die Support-Industrie derzeit (1996) verschwindend klein. Ihm sei nur eine Firma bekannt, Cygnus, die einen jährlichen Umsatz von mehr als einer Million Dollar hat. Deshalb erwartete er, daß mehr Autoren versuchen werden, in Form von Shareware oder einer Doppellizenzierung mit freien *not-for-profit*- und parallelen kommerziellen Versionen eine Entschädigung zu erhalten. Aus der Sicht eines kommerziellen Distributors sei freie Software kein besonders profitables Geschäft. Selbst ihre Leistung, Programmen ausfindig zu machen, könne durch immer ausgefeiltere Suchmaschinen überflüssig werden. Sein Fazit: "The rewards available to authors with pure FRLs [Freely Redistributable Licenses], and the nature of end-users' expectations, make it unlikely that functionally rich, end-user-oriented FRS will be created other than as shareware; however, the FRS model can work well for more developer-oriented applications. It will be interesting to see how FR and non-FR

---

[484] Lewis 7/2000

[485] Stefan Meretz, oekonux@mikrolisten.de, Tue, 14 Dec 1999 08:53:48 +0000, http://www.oekonux.de

[486] Deutsch 1996

software compete in the emerging domain of reusable components."[487] Die Bewegung ist inzwischen auf dem besten Wege, Deutschs erste Einschätzung zu widerlegen. Seine Frage nach den Komponentenarchitekturen bleibt weiter interessant.

Eine ähnliche Lagebeschreibung liefert im selben Jahr Jim Kingdon, der vorher für Cygnus und die FSF gearbeitet hatte und zu der Zeit bei Cyclic Software beschäftigt war.[488] Cygnus (heute Red Hat) mit damals 50 Beschäftigten und das kleinere Startup Cyclic (heute OpenAvenue[489]) waren die frühen Vorzeigebeispiel für erfolgreiche Geschäftsmodelle mit freier Software. Kingdon nennt außerdem das X Window-Firmenkonsortium, das damals etwa 25 Personen beschäftigte, und die FSF, für die acht Angestellte arbeiteten. Als Support-Dienstleister erwähnt er Yggdrasil und als Distributor Red Hat. "Five successful ways of making money are by providing custom work for ports and new features, support contracts, training, consulting/customization, and telephone support." Er beschreibt sie ausführlich und gibt Beispiele aus der Zeit. Seine Folgerung: "People in free software businesses can make a living. And free software businesses need not inhibit in any way the communities of users and contributors which have been so instrumental in many free software projects." Was die Endnutzer-Software betrifft ist Kingdon optimistischer als Deutsch: "Software for hackers [...] is only the beginning."

In den vier Jahren seither hat die freie Software eine gewaltige Dynamik entfaltet. Heute gibt es die unterschiedlichsten Hybridformen in einer wachsenden Grauzone zwischen Open Source und eCommerce. Auch *Venture Capital* und das schnelle Geld der Börsengänge ist im Spiel. Vor allem der begrifflichen Wende von 'Free Software' zu 'Open Source Software' folgte -- Raymonds Kalkül erfüllend -- eine Wende des Marktes. Zu den ersten Ereignissen gehörte noch 1998 der Deal zwischen IBM und der Apache-Gruppe, bei die IBM-Anwälte entsetzt fragten, wie sie denn einen Vertrag mit einer Website schließen sollen. Die Phase der Unschuld ist lange vorüber. Spätestens nach dem überaus erfolgreichen Börsengang von Red Hat im August 1999, strömt das Geld in alles, was einen Pinguin trägt. Im Dezember 1999 brach VA Linux Systems alle Rekorde, als die $30-Aktie am ersten Tag für ganze $300 gehandelt wurde (die Höchstmarke unter allen IPOs stand bei 606% am ersten Tag, VA trieb sie auf 697,50%). Im Januar 2000 zog TurboLinux Investitionen in Höhe von $50 Mio von Dell, Compaq, Toshiba, NEC u.a. an, nachdem zuvor bereits Intel zugeschlagen hatte. Intel hat ebenfalls in Red Hat und SuSE investiert. Caldera Systems bekam $30 Mio und meldete sein IPO an. LinuxCare bekam $32 Mio von Sun. Covalent, Support-Anbieter für Apache, hat eine Finanzspritze von $5 Mio erhalten und der kommerzielle Supporter Linuxcare eine von $30 Mio. Zu den Investoren gehören Dell und Oracle.[490]

All das kann sich natürlich mit den mehr als $6 Milliarden Einnahmen nicht messen, die Microsoft im selben Jahr erzielte. Doch, daß mit einem Mal hunderte Millionen Dollar in einen sozialen, technischen, kreativen Prozeß fließen, in dem Geld bislang nur eine sehr eingeschränkte Rolle spielte, wird ihn ohne Frage verändern. Die bislang zu beobachtenden Folgen waren ambivalent. Auf der positiven Seite ist sind Einrichtungen wie Sourceforge[491] von VA Linux und das Red Hat Community Center[492] zu nennen, mit dem Unternehmen der

---

[487] ebd.

[488] Kingdon 12/1996

[489] http://www.cyclic.com/

[490] c't, 1/2000

[491] http://sourceforge.net/

[492] http://www.redhat.com/apps/community/

freien Software der Community wertvolle Ressourcen zur Verfügung stellen. Problematischer scheint sich das Verhältnis zu traditionellen Software-Uternehmen zu gestalten, wie die gespannten Beziehungen zwischen Corel und KDE zeigen.[493]

"Wieviel Geld läßt sich mit freier Software verdienen?" Dieser Frage ging eine Studie des Investment-Dienstleisters WR Hambrecht + Co im Mai 2000 nach. Die Antwort: "Our estimate: over $12 billion by 2003. Revenue from the market for Linux products and services is expected to explode with a compounded annual growth rate (CAGR) of 90%, growing rom $2 billion in 2000 to over $12 billion in 2003. [...] While the revenue opportunity presented by companies focused on Linux products and services seems large, we believe that these estimates are only the beginning."[494] Vor allem mit der weiteren explosiven Ausbreitung des Internet und dem Anbruch einer mobilen und Post-Desktop-Ära sieht die Studie ein mögliches lawinenartiges Anwachsen des freien Software-Marktes voraus.

Im Folgenden werden die wirtschaftlichen Potentiale freier Software aus Sicht der verschiedenen beteiligten Aktoren betrachtet, den Anwendern, den Dienstleistern (Distributoren, Systemhäuser und *Application Service Providers*), den Autoren und der Handbuchbranche.

Anwender von freier Software

Hauptgewinner sind die Anwender, die den Gebrauchswert der Programme erhalten. Ob private Anwender oder Systemverwalter, sie können sich aus dem Pool freier Software selbständig bedienen und von den Vorteilen, wie Flexibilität, Stabilität (weithin getestet und debugged), Investitionssicherheit und Betriebssicherheit (Software läßt sich nur dann auf Sicherheitsmängel und Hintertüren überprüfen, wenn der Quellcode vorliegt), profitieren. Kooperative Hilfe finden sie im Internet.[495]

Freie Software bietet die Chance, sich von proprietären Anbietern zu emanzipieren und sich selbst zu qualifizieren. Sie fordert zum Ausprobieren und zum Lernen heraus. Dabei vermittelt sich duch Freie Software neben dem Wissen über die Anwendung auch das Wissen über die zugrundeliegende Technologie. Die Gängelung durch herkömmliche Industriepraktiken verunmöglicht nicht nur eigene Modifikationen, sondern kann schon die reguläre Anwendung eines Produktes schwierig machen, wie eine Erfahrung der taz zeigt: "Z.B. hatten wir das Problem, daß wir unser NT-Netzwerk vernünftig backupen wollten und feststellten, daß wir überhaupt keine vernünftige Dokumentation darüber bekommen, wie ich das jetzt eigentlich machen kann. Man muß ein *Non-Disclosure Agreement* unterschreiben und bekommt dann unter Umständen die Hinweise -- das weiß man vorher gar nicht genau. Und wenn man dann daraus etwas entwickelt, kann ich das noch nicht einmal weitergeben."[496] Dagegen erlaubt freie Software eine individuelle Anpassung und

---

[493] Als Corel im Sommer 1999 seine Debian-basierte Linux-Distribution mit KDE auslieferte, hatte es Namen von Applikationen geändert, das Aussehen soweit als möglich an Windows 95/98 angepaßt und seinem KDE ein Corel-Copyright verpaßt. Seither scheint seine Beteiligung am KDE-Projekt durch einen eigens dafür angeheuerten Interface-Designer darauf zu zielen, KDE in eine Corel genehme Richtung zu lenken. (Powell 6/2000a

[494] Patel 5/2000

[495] weitere Informationen und Success Stories auf den "Business"-Seiten von linux.de: http://www.linux.de/business/

[496] Klever, Wizards 7/1999

Weiterentwicklung durch eigene Mitarbeiter oder externe Dienstleister. Die Integration in bestehende Systeme und die Herstellung von Interoperabilität mit anderen freien oder kommerziellen Lösungen ist möglich.

Freie Software ist kostengünstiger. Einer Gartner-Group-Studie zufolge betragen die Kosten für Software-Lizenzen im Schnitt nur etwa 10% der Investitionen in Internetprojekte.[497] Doch auch wenn die Software gebührenfrei beschafft werden kann, ist die *Total Cost of Ownership* natürlich nicht gleich Null.[498] Cygnus, die erste Firma, die kommerziellen Support für GNU-Software anbot, warb mit dem Slogan "Making Free Software Affordable". Die Installation und Wartung kosten weniger als bei vergleichbaren kommerziellen Produkten. Wie in jede neue Software müssen sich die Anwender zunächst einarbeiten. Die Einstiegshürde ist hier z.T. größer als bei proprietärer Software (Beispiel: Sendmail), doch führt die Offenheit zu einem höheren Maß an Verständnis dafür, wie die Software arbeitet. Alle folgenden Anpassungen und Entwicklungen sind daher mit weniger Aufwand verbunden. Bei einem Wechsel der Hardware oder der Plattform können die erworbenen Kenntnisse weiterverwendet werden. Das bedeutet eine Unabhängigkeit von den Herstellern und damit eine hohe Investitionssicherheit. Ein weiteres Beispiel aus der Erfahrung der taz:

> "Ich muß nicht, wenn ich eine Plattform oder das Betriebssystem-Release wechsle, eine neue Lizenz vom Software-Hersteller kaufen oder habe Beschränkungen, wie sie z.B. bei Sun-Maschinen sehr üblich sind, daß sie abhängig sind von einem *Motherboard* und die Software nur auf dem *Board* läuft. Wenn der Rechner kaputt geht, habe ich ein Problem und muß erst einmal zwei, drei Wochen hin und her faxen, bis ich eine neue Lizenz bekomme. Oder ein anderes Beispiel: Ein Hersteller von ISDN-Routing-Software teilte uns mit, als wir von Sun-OS auf Solaris umgestiegen sind, daß wir unsere gesamten ISDN-Karten neu kaufen müßten, weil die alten Karten nicht mehr von ihrer Software gepflegt werden. Dabei ging es um Investitionskosten in Höhe von 100.000 DM. Da hätte ich, wenn ich die Sourcen gehabt hätte, gerne jemanden darauf angesetzt, der das für mich analysiert und dann die entsprechende Anpassung gemacht hätte."[499]

Durch ihre Unabhängigkeit von Produktzyklen, Marktkonzentration und Insolvenzen bietet die freie Software eine unübertreffliche Investitionssicherheit. Die kontinuierliche Entwicklung kann durch immer neue Generationen von Programmierern aufrecht erhalten werden.

Die Angst vor der Abhängigkeit von einem einzigen Lieferanten einer Software ist für viele Anwender in Unternehmen und Behörden ein wichtiges Argument für freie Software. Sie schließt Monopolbildung aus und fördert eine lokale KMU-Infrastruktur von Dienstleistern (allerdings schließt sie einen Support durch weltweite Unternehmen wie IBM nicht aus). Während sich die Hardware-, Software- und Internet-Industrie in den USA konzentriert, kann das offene Kooperationsmodell auf Basis des Internet unvergleichlich mehr kreative und produktive Ressoucen erschließen. "Die große Beliebtheit von OSS in

---

[497] Gartner-Group nach Gessner, Wizards 7/1999

[498] für Beispielrechnungen s. Broschüre des BMWi zu Open Source Software, erscheint voraussichtlich im November 2000

[499] Klever, Wizards 7/1999

Deutschland und die hohe Zahl der OSS-Entwickler und -Entwicklungen in Europa sind ein deutliches Zeichen dafür, daß die innovative Kraft überall vorhanden ist. Die Zukunft der Informationstechnologie liegt nicht in einem kalifornischen Tal sondern in den Weiten des Internet."[500]

Ferner sind Anwender zu nennen, die auf Grundlage freier Software neue Märkte gründen, z.B. das *Internet Service Provider*-Geschäft. Rick Adams, ehemals Betreiber des größten Usenet-Knotens der Welt und Autor von BNews, der am weisten verbreiteten Usenet-News-Software, gründete UUNET:

> "And he really invented what we now take for granted, the commercial Internet Service Provider Business. So when people think about free software and money, they think, they very often play right in to Bill Gates' hand because they think about the paradigm that he has perfected so well, which is put software in a box, ship it, get a locked in customer base, upgrade them. And Rick just kind of went sideways from that. And he was the first person to say, 'I'm gonna build a serious business that is based on free software,' and it was basically providing a service that was needed by the people who used that software, who talked to each other, who distributed it, who worked with each other online."[501]

Eine Netzgeneration später waren es Firmen wie Amazon und Yahoo, die eine neue Klasse von 'Infoware'-Dienstleistung[502] auf Grundlage freier Software entwickelten.

Kein Wunder also, daß GNU/Linux auch im Management immer mehr Akzeptanz gewinnt. Die WR Hambrecht-Studie schreibt: "According to an Information Week research survey of 300 IT managers, Linux use by application is migrating from handling Web servers and email to areas such as systems management, thin servers, and even enterprise resource planning. This data is significant because it shows that Linux is truly gaining legitimacy as it is being deployed in critical enterprise environments."

Dienstleister rund um freie Software

Mit der wachsenden Popularität freier Software steigt auch die Nachfrage nach Dienstleistungen wie Beratung, Installation, Support, Wartung, Schulung und Gewährleistung. In den Unternehmen, in denen bislang die hausinternen Techniker unter der Hand freie Systeme eingeführt hatten, wächst seit 1998/99 die Akzeptanz auch im Management. Damit entsteht eine Nachfrage nach externen Dienstleistungen, z.B. der Entwicklung maßgeschneiderter Systemlösungen für Kunden und der Auftragsentwicklung neuer freier Software. Linux-biz.de listet fast 150 Firmen in ganz Deutschland, die sich auf Dienstleistungen für GNU/Linux spezialisiert haben.[503] Auch die Tatsache, daß Unternehmen wie IBM ein flächendeckendes Service-Angebot für Linux aufbauen, läßt darauf schließen, daß hier erhebliche Marktchancen gesehen werden.

Tatsächlich machen Dienstleistungskosten den Löwenanteil im IT-Bereich aus. Eine Gartner-Group-Studie über *Software-Investments* von 1998 ergab, daß ungefähr 80% aller

---

[500] Hetze 1999: 9

[501] O'Reilley, Wizards 7/1999

[502] s.o. unter "Geschichte der Software-Entwicklung"

[503] http://linux-biz.de/firmen.html

Internet-Investitionen in den Service (*Professional Services, Consulting, Tech-Support* usw.) fließen und nur 20% zu gleichen Teilen in das Anlagevermögen, also Hardware und Software.[504]

Patrick Hausen vom BSD-Projekt erläutert den Zusammenhang von unbezahlter Entwicklung in den Projekten und Dienstleistungsbranche:

> "Ich habe vielleicht zehn Entwickler, die diese Apache-Software für Gotteslohn schreiben. Ich habe hunderttausend Anwender. Dann brauche ich für diese hunderttausend Anwender tausend Consultants, von denen jeder hundert Anwender betreuen kann. D.h., ich habe tausend Consultants, die davon leben können, und ich habe hunderttausend Anwender, die durch den Einsatz dieses kostenlosen Webservers mehr Geld für vernünftiges Consulting ausgeben können und letztendlich ja auch wieder etwas tun mit dieser Software, was hoffentlich in der einen oder anderen Form Umsatz generiert."[505]

Über den Dienstleistungsmarkt äußert sich die WR Hambrecht-Studie am optimistischsten: "Perhaps the most significant opportunity in the Linux market for Linux services, support, and training. We project revenue of almost $500 million in 2000, growing at a CAGR of 100% to nearly $4 billion in 2003. Although traditional companies like IBM and Hewlett Packard (HWP) have begun to provide Linux consulting and support, this market opportunity is largely untapped."[506]

Systemhäuser, Hard- und Software-Hersteller

Anbieter von proprietärer Software und von Hardware setzen freie Software als Basistechnologie ein. Für Cobalt Micro war Linux der Schlüssel, mit dem es den Servermarkt für eine neue Hardwareplattform öffnen konnte. SGI ersetzt Irix durch Linux und hofft auf diese Weise Synergieeffekte nutzen zu können. Zu den Firmen, die Linux-Hardware vertreiben, gehören VA Linux, Cobalt, Atipa, aber auch traditionelle Unternehmen wie IBM, Compaq und Dell. Die WR Hambrecht-Studie kommt zu der Einschätzung: "Linux hardware companies sell the crucial systems that serve the growing number of the world's Web pages, email, and internal information networks. We expect revenue from Linux hardware to grow from almost $1.2 billion in 2000 to over $7.7 billion in 2003, with a CAGR [compounded annual growth rate] of 83%. [...] The Linux software market is in the early stages of its growth and maturity. Revenue from the sale of Linux client and server operating system (OS) software is predicted to grow from $160 million in 2000 to over $700 million in 2003, with a CAGR of 67%."[507]

Ein Marktsegment, das heute noch in den Kinderschuhen steckt, auf das sich aber große Erwartungen richten, ist das für eingebettete Systeme. Die Studie, die diese Entwicklung noch ausklammert, sieht hier ein Potential, das den gesamten derzeitigen Linux-Markt als winzig erscheinen lassen könnte.

---

[504] Gartner-Group nach Gessner, Wizards 7/1999

[505] Hausen, Wizards 7/1999 Diskussion

[506] Patel 2000

[507] Patel 2000

Auf ähnliche Weise ist freie Software für Anbieter von standardisierten Branchenlösungen interessant. Die Betriebssystemplattform und die Software-Umgebung sind für die Kunden unwesentliche aber eventuell kostenrelevante Nebenleistungen des eigentlichen Geschäfts. Hier kommen ebenfalls Linux und die freien BSD-Derivate zum Einsatz.

Aus demselben Grund bieten sie sich auch als Basis für kundenspezifische Softwareentwicklung an. Software-Häuser finden dazu eine Fülle wiederverwendbarer Bausteine und können unter Umständen sogar aus dem kooperativen freien Entwicklungsprozeß Nutzen für ihr jeweiliges Projekt ziehen.

Distributoren

Die vorangegangenen Wissensordnungen hatten ihre Materialität und ihre Leiblichkeit. Datenträger (also Bücher, zumindest bis zur Erfindung neuer Papier- und Drucktechniken Ende des 19. Jahrhunderts) und gute Lehrer waren Mangelware. Das digitale Wissen hat natürlich immer noch einen physikalischen Träger, doch innerhalb des Internet kann es -- der Sache nach -- frei fließen. Wissen gibt es im Überfluß.

Die Verbreitungskosten gehen bei Bereitstellung auf einem Server zum freien Download gegen Null. Ob von einer Software zehn oder zehn Millionen Kopien gemacht werden, spielt keine Rolle. Proprietäre Software dagegen muß eine artifizielle Knappheit herstellen, indem sie den Kopierprozeß monopolisiert. Ein Teil der Kosten für die Erzeugung der Knappheit (z.B. polizeiliche und gerichtliche Maßnahmen gegen ‘Raubkopierer’) werden externalisiert.

Distributionen sind Sammlungen freier Software auf CD-ROM (früher auf Disketten). Sie ersparen dem Anwender die Recherche (Was ist die neueste Version eines Programms? Was ist die letzte stabile? Arbeiten bestimmte Programme zusammen?) und die Übertragung von z.T. sehr großen Datenmengen aus dem Internet. Firmen, die solche Sammlungen vertreiben, bieten zusätzlich ein Installationsprogramm, Dokumentation, Verpackung und Installations-Support. Neben den Linux-Distributoren wie RedHat und SuSE bieten Walnut Creek oder PrimeTime Freeware Sammlungen freier Software an.

O’Reilly führt den Gründer und CEO von Red Hat an: “Bob Young likes to say: ‘We're not a software company, we're a sales, marketing and distribution company, just like Dell.’ Now whether or not Red Hat in fact is the winner in what's shaping up to be the Linux distribution wars and probably the next big Wall Street feeding frenzy is somewhat irrelevant. The fact is that there is a business model niche.”[508]

Auch Hersteller von proprietärer Software wissen den nahezu kostenlosen Distributionseffekt für sich zu nutzen. Corel gibt sein Word Perfect und Star Division (und nach dessen Aufkauf Sun) sein StarOffice für persönlichen Gebrauch kostenlos ab, während Corel sich den Einsatz in Unternehmen und Behörden bezahlen läßt. Das hat zwar nichts mit *free software* und mehr mit *free beer* zu tun, zeigt aber, daß ein anderes Business-Modell dazu führen kann, daß der de facto Status (freies Kopieren im Privatbereich) legalisiert wird. Selbst Microsoft ist sich bewußt, daß eine gewisse Durchlässigkeit der Copyright-Regeln ihm mehr nützt als schadet und bedient z.B. den schwarzen Markt in Indien mit *Not For Resale*-Kopien seiner Software.[509]

---

[508] O’Reilly, Wizards 7/1999

[509] persönliche Kommunikation mit Atul Chitnis, Exocore Consulting, Bangalore, Indien

Application Service Providers (ASPs)

Ein neues Paradigma in der Informatik sind *thin Clients*, die ihre Anwendungen und Daten von zentralen Servern beziehen. Solche Strukturen können sowohl im unternehmensinternen LAN, wie Internet-weit errichtet werden. Im zweiten Fall hat ein Nutzer von überall Zugriff auf seinen 'Arbeitsplatz im Netz.' Dafür sollen monatliche oder *per use*-Gebühren für die Nutzung und ggf. Mietgebühren für den verwendeten Plattenspeicher anfallen.

Sun Microsystems ist ein Vorreiter dieser Bewegung und hat das Hamburger StarDivision aufgekauft, um dessen Hauptprodukt StarOffice als APS-Anwendung anzubieten. Im Juli 2000 stellte Sun den über 15 Jahre entwickelten Quellcode des Office-Pakets unter die GPL.[510] Offenkundig ist es Sun also wichtiger, den *Mindshare* der freien Entwicklerszene für das Projekt zu gewinnen, als aus der kontrollierten Vermietung von StarOffice Profit zu schlagen. Die wird es in erster Linie aus dem Verkauf von Thin-Client-Hard- und Software beziehen.

Das gleiche *Server-Side Computing* Modell betreibt auch das im Januar 1999 gegründete Workspot, das einen Linux-Desktop im Netz anbietet.[511] Mit einem Webbrowser, einem PDA oder einem WAP-Telefon kann man sich von überall in der Welt in seinen persönlichen Arbeitsplatz einloggen, den man -- mit allen geöffneten Applikationen und Daten -- so vorfindet, wie man ihn beim letzten Mal verlassen hat. Die professionelle zentrale Systemwartung inkl. Backups bannt die Gefahr für Daten durch defekte Festplatten und entwendete Laptops. Das System befindet sich noch in der Testphase und soll durch die Vermietung von unfreier Software und Plattenplatz, sowie Hosting von öffentlicher Software oder Intranet-Lösungen Dritter Gewinne einspielen. Zu Workspots Kunden gehören eBay, Barnes & Noble und OmniSky. Workspot hebt hervor, daß es auch eine Gelegenheit für Linux-Interessierte sei, es auszuprobieren, ohne sich erst lokal Software installieren zu müssen. Aus demselben Grund sei es eine attraktive Plattform für Entwickler, die ihre Software hier zum Ausprobieren bereitstellen könnten.

> "Making this dream a reality involves distributed computing, virtual machines and extending the use and migration of persistent data. These are technological challenges that could not be met on proprietary systems.
>
> Enter Linux, and free software. The source code is available for modification and improvement until the end of time. It's a grand, busy, community-minded movement. And it's the best environment for building large, coherent systems. Server-side computing is natural for Linux, and will naturally drive Linux [to] become the best consumer computing experience."

Projekt-Hosting und Portale

Ein weiteres neues Business-Modell ist direkt aus dem Kooperationsmodell der freien Software hervorgegangen. Unternehmen wie Andover.Net (das im Juni 1999 das wichtige Szene-Forum slashdot.org kaufte) und Collab.Net (ein Spin-Off von O'Reilly & Associates, gegründet und unter Präsidentschaft von Brian Behlendorf, Mitgründer von Apache[512])

---

[510] s. http://openoffice.org

[511] http://www.workspot.com

[512] Scanlon 11/1999

betreiben seit 1999 Sites, auf denen Käufer und Anbieter von Open Source-Projekten einander finden können. Unternehmen schreiben hier gleichsam Programmieraufträge aus und wieviel sie dafür bereit sind zu bezahlen, und freie Enwickler und Firmen können sich darum bewerben. Andover.Nets *QuestionExchange*[513] und eBay Open Source verwenden dafür eine Auktionssystem. Collab.Nets *sourceXchange*[514] und EarthWeb hosten freie Projekte, bieten Marktplätze für Support und hosten *gated Communities*. Profite machen diese Firmen mit Werbeeinnahmen und mit Transaktionsgebühren. Die Einschätzung der WR Hambrecht-Studie ist auch hier rosig:

> "Successful Linux Internet properties will leverage Web traffic into increased opportunities in advertising, ecommerce, and business-to-business development services and support. [...] Revenue from online Linux advertising is projected to grow from $33 million in 2000 to nearly $500 million in 2003, a CAGR of 132%. These estimates only include revenue from online advertising and we anticipate that other significant revenue opportunities will emerge, such as online Open Source markets. While the Linux advertising revenue numbers may not seem significant, it is important to recognize that revenue from online advertising is very high margin (typically greater than 70% gross margin)."[515]

Erstellung freier Software

Auf die Motivelagen freier Entwickler wurde oben bereits eingegangen. Es hatte sich gezeigt, daß hier kreative, kulturelle, soziale und Lernaspekte an die Stelle einer pekuniären Entlohnung treten. Auch hierin steht die Bewegung in einem größeren Trend zur Neubewertung der sozialen, kulturellen, kreativen Arbeit außerhalb der Lohnarbeit, wie Szyperski sie benennt:

> "Wir leben, oberflächlich betrachtet, in der sogenannten Freizeitgesellschaft. Wenn Sie sich mal ein Jahr vorstellen, dann haben wir 8760 Stunden zur Verfügung. Die Erwerbstätigkeit in Deutschland macht nur noch 1900 Stunden aus -- im Gegensatz zu Japan, wo es etwa noch 2300 sind, aber irgendwo um diese Größe pendelt sich das in der Welt ein. Wenn Sie nun davon ausgehen, daß wir zehn Stunden am Tag Schlaf, Ruhe, Speisen, Pflege brauchen, dann bleiben insgesamt 3210 Stunden im Jahr übrig. Was ist das für eine Zeit? Wollen wir die jetzt wieder in die Erwerbszeit integrieren? Wollen wir sie nur in die Schlaf-, Pflege- und Entertainment-Zeit integrieren? Oder -- und das ist ein Vorschlag, der sehr intensiv z.B. von Amitai Etzioni im Zusammenhang mit der Reaktivierung des Kommunalverständnisses und der kommunalen Aktivitäten diskutiert wird --, wollen wir die Zeit investieren, um für unsere Gemeinschaften konstruktiv Beiträge zu leisten, die nicht in irgendeiner Form über Geld abgerechnet werden? Das sind Sozialleistungen, das sind Sportleistungen, das ist aber auch die Selbstverwaltung unserer Organe, das ist das Lehren und Lernen außerhalb der geordneten Schulen, das ist aber auch das Erziehen der Kinder, das ist

---

[513] http://www.questionexchange.com

[514] http://www.sourcexchange.com

[515] Patel 2000

> das Machen der Familie und ich weiß nicht was alles. Das heißt, wenn wir so einen großen Teil der verfügbaren Zeit ‘Freizeit’ nennen, dann mißdeuten wir schon ihre Funktion. Darum gibt es Bemühungen, das als ‘Sozialzeit’, ‘Gemeinschaftszeit’ oder wie auch immer -- mir ist da jedes Wort lieb -- zu bezeichnen.”[516]

Neben der unbezahlten *Volunteer*-Arbeit gibt es auch Firmen, die sich auf die Kommerzialisierung freier Software spezialisieren. Das mehrfach genannte Cygnus Solutions war eine der ersten. 1989 wurde es in Californien gegründet und bot vor allem Hardware-Herstellern an, die GNU-Tools auf ihre Plattform zu portieren. Die Dienstleistung wird bezahlt, das Ergebnis ist, wie es die GPL vorschreibt, frei. Es schuf mehr als 120 *Host-Target*-Kombinationen seiner Werkzeuge für eingebettete Mikroprozessoren und bot Auftragsprogrammierung, Entwickler-Support und andere Leistungen. Die Fima hatte Vertretung in Nordamerika, Japan und England und wurde seit 1997 in Software Magazines List der führenden 500 Software-Unternehmen aufgeführt.[517] Cygnus wurde im November 1999 von Red Hat für $674 Millionen aufgekauft.[518]

Zahlreiche weitere Firmen sind dem Vorbild gefolgt und setzen entweder vollständig oder in einem Teil ihres Angebots auf freie Software. Sie entwickeln Programme aus bestehender freier Software oder integrieren Bestandteile freier Software in ihre Programme, die aus lizenzrechtlichen Gründen ihrerseits frei sein müssen. Die Einbindung von Bibliotheken in proprietäre Software ist nach dem Open Source-Modell zulässig, nach dem FSF-Modell jedoch allenfalls dann, wenn sie unter der LGPL stehen. Auch freie Compiler wie der GCC, und andere Tools, wie CVS oder FreeCASE, werden in der kommerziellen Software-Entwicklung eingesetzt. In dem Fall kommt es nicht zu einer Lizenzvererbung, d.h. die entstehenden Produkte dürfen proprietär sein.

Entsprechend gewachsen ist der Arbeitsmarkt in diesem Bereich. Dem kommt entgegen, daß freie Software häufig in Forschung und Lehre eingesetzt wird, Informatiker also meist fundierte Erfahrungen mitbringen. Und um Entwickler, die sich in der freien Szene verdient gemacht haben, reißen sich die Unternehmen ohnehin.

Schließlich gibt es noch Unternehmen wie Sun, Netscape und IBM, die ihre konventionell von ihren Angestellten entwickelte Software quelloffen machen und mit Arbeitskraft aus ihrem Haus weiterentwickeln. Netscape erläutert, welchen Gewinn es sich davon verspricht:

> “Netscape (and the world) benefits from giving away its source code in a number of different ways. First and foremost is from the expected explosion of creative input to the Communicator source code base. Netscape is confident that this will allow Communicator to retain its position as the preeminent communications software. Second, freeing the source code allows it to go places and into products that Netscape may never have had the resources or time to take it on its own, extending internet accessibility to hardware platforms and software products far and wide.”[519]

---

[516] Szyperski, Wizards 7/1999

[517] Tiemann 1999

[518] http://www.redhat.com/about/cygnus_1999/redhat-cygnus111599.html

[519] Netscape Public License FAQ, 25.How does Netscape benefit from giving away its source code under such a free license?, http://www.mozilla.org/MPL/FAQ.html

Die WR Hambrecht-Studie stellt fest: “The market for Linux applications has yet to emerge. However, we feel that the opportunity from proprietary software companies opening the source code to some or all of their product offerings could be a significant, disruptive event in the software industry. The result could be software that is more reliable and amenable to the needs and utility of end users.”[520]

Dokumentation

Auch diejenigen, die sich die Software aus dem Internet beschaffen oder (durch die Lizenzen zulässig) CDs von Freunden ausleihen oder kopieren, benötigen in der Regel Handbücher und andere Fachliteratur. Marktführer bei solchen Handbüchern für freie Software ist der O’Reilly Verlag (s.u. unter “Praxisbeispiele”). O’Reilly schätzt, daß er im Laufe der Jahr Bücher über Open Source-Software im Wert von mehr als 100 Mio Dollar verkauft hat und sein Open Source-bezogener Umsatz im vergangenen Jahr größer war, als der von SuSE oder Red Hat.

In Deutschland ist der Fachbuchhandel für EDV-Literatur auch zu einem wichtigen Vertriebskanal für die CD-ROM-Distributionen selbst geworden. Hier hat sich vor allem die Lehmanns Buchhandelskette einen Namen gemacht (s.u. unter “Praxisbeispiele”).

[520] Patel 2000

Praxisbeispiele

Firmen, die sich mit freier Software beschäftigen, lassen sich grob in drei Kategorien einteilen (auf die Beispiele in den Klammern wird im Folgenden näher eingegangen):

1. Anbieter von freier Software und Dienstleistungen (SuSE, Lunetix, innominate, New Technologies Management GmbH)
2. Anwender von freier Software (Lehmanns Buchhandlung, die tageszeitung, Babcock-BSH, Individual Network)
3. Anbieter von quelloffener, aber im Sinne der GPL unfreier Software und von proprietären Produkten für freie Plattformen, in den meisten Fällen GNU/Linux und BSD, aber auch Module für Apache etc. (Apple, O'Reilly Verlag, Intershop).

Viele der freien Projekte haben keinerlei Bedenken dagegen, daß ihre Software kommerziell verwertet wird. Auch hier setzte die FSF Zeichen, indem sie von Beginn an einen Teil ihrer Arbeit aus dem Verkauf von Distributionen auf Datenbändern und dann CDs finanziert. Dennoch besteht ein Spannungsfeld zu denjenigen, die den Gelderwerb durch Dritte ablehnen, das z.B. auf der Linux-EXPO in Durham 1998 zur Sprache kam. Doch "die einzigen, die sich wirklich aufgeregt haben, waren Reporter, die meinten, da muß doch ein Konflikt sein."[521] Im Gegenteil freuen sich viele Entwickler, wenn Firmen ihre Software verkaufen und die Verbreitung damit erhöhen. Dankbar sind sie dafür, daß Distributoren das Packaging, die Produktion von Handbüchern und den Support übernehmen und so die Entwickler entlasten, die sich dadurch auf das konzentrieren können, was ihnen Spaß macht: das Entwickeln.

LunetIX

Am Anfang stand das "Linux Anwenderhandbuch" (LHB), das heute immer noch, inzwischen in der 7. Auflage, das deutschsprachige Standardwerk ist. Als Sebastian Hetze, Dirk Hohndel, Olaf Kirch und Martin Müller 1992 seine erste Version fertiggestellt hatten, fand sich dafür kein Verlag, da noch niemand von Linux gehört hatte. So gründeten sie ihren eigenen Verlag, und ein Jahr später stand das LHB in den Bestsellerlisten der Computer-Literatur vor Titeln zu Microsoft-Word und Excel. Das ist umso bemerkenswerter, da das LHB parallel zu gedruckten Version im Netz frei verfügbar[522] und unter den Bedingungen der GPL für nichtgewerbliche Zwecke frei kopierbar und weitergebbar ist. So sind viele der LHB-Texte in die Online-Dokumentation (die *man pages*) deutschsprachiger GNU/Linux-Distributionen, wie der von SuSE, eingegangen. Wie SuSE ist die LunetIX Müller& Hetze GbR[523] 1992 gegründet worden. Aus dem Verlag wurde schnell ein Linux-Systemhaus. Noch im selben Jahr stellte es seine erste Linux-Distribution, die LunetIX Software-Distribution (LSD) zusammen und vertrieb sie über die Lehmanns-Buchhandlungen und kurz darauf auch über eMedia des Heise Verlags.

LunetIX bot auch von Anfang an Support für Linux an, doch gab es in den ersten Jahren dafür fast keinen Bedarf. Zwar gab es auch in dieser Zeit bereits Unternehmen, die

[521] Hohndel, Wizards 7/1999 Diskussion

[522] http://www1.lunetix.de/LHB/

[523] http://www.lunetix.de

Linux einsetzten, doch waren es in der Regel firmeninterne Systemadministratoren und Entwickler, die Projekte mit freier Software realisierten. Diese Art von *bottom-up* Verbreitung bedeutete, daß keine Etats bereitstanden, um Dienstleistungen von außen einzukaufen. Die Praktiker in den Unternehmen mußten sich selbst behelfen und konnten das auch dank der Unterstützung, die sie im Internet fanden.

Auch heute bietet LunetIX noch Support an, nur daß die Popularität von Linux inzwischen auch das Management erreicht hat, das jetzt entsprechende Etats für externe Dienstleister vorsieht. Im Frühjahr 2000 wurde LuneIX zum Teil der Linux Information Systems AG.[524] Während Hetze im reinen Distributionsgeschäft nur begrenzte Entwicklungsmöglichkeiten sieht, hält er die individuelle Anpassung auf Kundenerfordernisse, die Neuentwicklung im Kundenauftrag, Wartung, Schulung und andere Dienstleistungen für die erfolgversprechensten Geschäftsmodelle für freie Software. Sie seien gegenüber proprietärer Software für den Kunden günstiger, "weil wir die Wiederverwendung von freier Software mit einkalkulieren können. Zudem ist es möglich, unter bestimmten Umständen zusätzliche Ressourcen durch kooperative Entwicklung, die von mehreren Unternehmen getragen wird, und auch von freien Entwicklern -- wenn man in der Lage ist, daraus ein richtiges Open Source-Projekt nach dem Basar-Stil zu entwickeln --, einzubinden und weiterhin für den Kunden Kosten zu sparen."[525]

Hetzes Vision für die Zukunft der freien Software in Deutschland ist, daß sich in dieser Zeit, die dafür reif ist, eine Vielzahl kleiner Firmen und selbständiger Entwickler herausbildet, die untereinander kooperieren können, aber auch mit dem Basar der freien Szene, um auch große Projekte im industriellen Maßstab durchzuführen. "Das ist mein Entwicklungsmodell, meine Vorstellung für eine Integration von wirtschaftlichem Arbeiten mit Open Software und der Entwicklung von Open Source-Software, die an erster Stelle eben nicht mit kommerziellen Interessen verbunden ist."

SuSE

Die Firma ist 1992 von vier Studenten gegründet worden. Der damalige Tätigkeitsschwerpunkt war Auftragsprogrammierung, also das, was man heute Neudeutsch als *Outsourcing* bezeichnet. Im Frühjahr 1993 kam SuSEs erste Linux-Distribution auf den Markt. Damit ist die SuSE einer der ältesten Anbieter kommerzieller Linux-Distributionen weltweit. Das US-amerikanische RedHat hat ungefähr ein Jahr später angefangen. SuSE hatte im Juli 1999 zirka 150 Mitarbeiter weltweit. Davon arbeiten gut 60% in der Technik, im *Support* und im Servicebereich.

Im Geschäftsjahr zwischen April 1998 und März 1999 hat SuSE zirka 14 Millionen Dollar Umsatz gemacht und ist damit der größte Umsatzmacher überhaupt weltweit im Linux-Umfeld. Die Firma ist seit Beginn ihres Bestehens von Jahr zu Jahr um etwa hundert Prozent gewachsen. Der allgemeine Linux-Hype trägt dazu bei, daß viele Firmen, darunter auch SuSE, stark expandieren. Im Spätherbst 1997 ist im Zusammenhang mit der Gründung der SuSE Inc. Niederlassung in Oakland, Kalifornien, die SuSE Holding AG etabliert worden. Im April 1999 sind die SuSE München GmbH (Vorstand: Harald Milz) und die SuSE Rhein-Main AG (Vorstand: Dirk Hohndel) gegründet worden. Unter der Holding stehen außerdem das alte Stammhaus, die SuSE GmbH in Nürnberg, der Verlag SuSE Press

[524] http://www.linux-ag.de

[525] Sebastian Hetze, Wizards 7/1999

und weitere Firmen, wie eine Werbeagentur, die in erster Linie PR-Kontakte herstellt. Schließlich wurden im Juli 1999 die SuSE-Labs eingerichtet. Sie haben das innovative Ziel, *Third Level Support* durchzuführen, d.h., technischen Support und *Bug-Fixes* auf Source Code-Ebene, und zwar mit garantierten Reaktionszeiten, 24 Stunden x 7 Tage und das ganze Jahr hindurch. Das Angebot richtet sich, weil es nicht kostenlos sein kann, natürlich in erster Linie an Großkunden, an Firmenkunden und an Kooperationspartner.

Die Käufer der normalen SuSE-Linux-CDs erhalten 60 Tage freien Installations-Support. Wer darüber hinausgehende Unterstützung benötigt, kann außerdem einen Wartungsvertrag abschließen.

Open Source bedeutet für SuSE, daß alle Software, die von der Firma entwickelt wird, wo immer es irgend möglich ist, an die Entwicklergemeinde zurückfließt. Dem stehen nur Vertraulichkeitsvereinbarungen mit Dritten, meist Hardware-Herstellern, entgegen. Beispiele für solche NDAs finden sich bei X-Servern, ISDN-Kartentreibern, KDE und ALSA (Advanced Linux Sound Architecture[526]), also Programme, bei denen die Hersteller der Meinung sind, daß die Information über ihre Hardware, die in der Software steckt, nicht an die Öffentlichkeit und damit an ihre Konkurrenten gelangen sollte. Solche Software wird ohne Quellcode in die Distributionen aufgenommen. SuSE fördert Open Source-Projekte, auch solche, die außerhalb der SuSE stattfinden und solche, die in Kooperation mit Firmen durgeführt werden, finanziell oder durch andere Ressourcen. Die engen Kontakte in die Entwickler-*Community* ermöglichen eine schnelle Umsetzung neuer Idee und schnelle Verfügbarkeit von *Bug-Fixes*.

Harald Milz sieht die Zukunft der Computer-Industrie darin, daß die Hardware immer billiger werde, bis sie zur *Commodity*, zur Massenware geworden ist und es möglich sei, die Hardware bei einem Kundenprojekt oder beim Kauf eines Rechnersystems fast kostenlos dazuzuliefern. Die Software werde nach seiner Auffassung den gleichen Weg gehen. Zumindest die Betriebssystem-Software und ein großer Teil der *Middleware* und der Systemadministrationswerkzeuge werde durch den Preisdruck und durch die Stückzahlen immer billiger. Open Source sei somit die einzig mögliche Strategie auf dem IT-Markt.

SuSEs Kernkompetenz liegt in der Weitereinwicklung von Linux. Die Firma engagiert sich aber auch in der Weiterentwicklung der Hardware, z.B. mit dem *SALT-Cluster*.[527] SuSE vertreibt als OEM- (*Original Equipment Manufacturer*) oder VAR- (*Value Added Reseller*) Produkte Komplettsysteme auf Rechnern von Herstellern wie IBM, Compaq, Siemens, SGI und anderen, die normalerweise mit einem Support-Vertrag dieser Hersteller verkauft werden.

Der zweite Wachstumsmarkt, auf dem SuSE agiert, ist der Dienstleistungsbereich. Die Firma will ihr Support- und Service-Angebot regional und international ausweiten. Auch die strategischen Kooperationen mit großen Hardware- und Software-Herstellern sollen ausgebaut werden. SuSE baut ferner einen eigenen VAR-Kanal auf, d.h., Systemhäuser, die in Deutschland und außerhalb bereits etabliert sind, sollen als zertifizierte Vertriebspartner gewonnen werden, um mit den SuSE-Produkten in die Fläche gehen zu können. Ein Kunde z.B. in einer Kleinstadt, kann auf diese Art davon ausgehen, daß er in Rufweite jemanden hat, der ihm bei einem Problem helfen kann.[528]

---

[526] http://www.alsa-project.org/

[527] http://www.suse.de/de/hardware/suse_hw/cluster/

[528] Harald Milz, Wizards 7/1999

Innominate
Die Linux-Dienstleistungsfirma wurden im Mai 1997 gegründet. Die beiden Gründer waren Informatikstudenten. Im Juli 1999 arbeiteten dort ca. 15 Techniker. Im April 1998 wurde Innominate Berliner Landessieger im "Startup-Wettbewerb", einem Gründungswettbewerb von Stern, Sparkassen und McKinsey. Im Mai 1998 wurde die Firma der europaweit erste Support-Partner von RedHat Software. Sie war auch die erste deutsche Linux-Firma, die *Venture Capital* bekam, nämlich von der BMP AG. Damit will Innominate weitere Standorte innerhalb Deutschlands eröffnen.

Innominate versteht sich als *Allround*-Systemhaus für Linux und BSD und bietet die gesamte Palette von Dienstleistungen an. Zur Beratung dient das Infotelefon 0180 1LINUX-BIZ und die Website www.linuxbiz.de, die die Firma zusammen mit dem LIVE Linux-Verband gestartet hat. Hier können sich kommerzielle Linux-Anbieter kostenlos eintragen lassen, und der Kunde kann sehen, wo er in seiner Nähe Linux-Support findet.

Schulungen von Standard-Einführungen bis zu maßgeschneiderten Schulungen werden bei Innominate selbst, als auch direkt beim Kunden durchgeführt. Kundenspezifische Lösungen werden entwickelt, angepaßt und implementiert. Das Support-Konzept ist abgestuft: eine kleine Firma, die nicht so schnelle Reaktionszeiten benötigt, kann einen kostengünstigeren Support erwerben, als eine größere Firma, die sofort jemanden braucht, der vor Ort hilft.

Zu der bei Innominate entwickelten Software gehört der Lingo-Kommunikations-Server auf Linux, der kleinen und mittleren Unternehmen, die kaum IT-Know-how im Haus haben, eine kostengünstige, stabile Lösung für Drucken, Faxen, Internet-Zugang, Email, sowie Datenbank- und File-Server-Anbindung bietet.

Zunehmend sind auch mittlere und große Firmen an innominate herangetreten, die Linux-Support und Linux-Solutions wollen, wie Netzwerkaufbau und Netzwerkmigrierung.

> "Für diese Firmen ist der Kostenfaktor bei der Anschaffung der Software überhaupt kein Thema mehr. Sie interessiert die *total cost of ownership*, d.h. was kostet mich die Software im Laufe des Einsatzes, und da ist der Kaufpreis nur noch ein verschwindend geringer Bestandteil. Diese Firmen sehen, daß Linux sehr wartungsarm und sehr administrationsarm ist und damit Kosten eingespart werden können. Diese Firmen sind an der Offenheit des Systems interessiert, an der Reaktionsschnelligkeit, wenn es Probleme gibt, wenn Fehler auftauchen und natürlich an der Sicherheit, die Linux einfach auch mit bietet."[529]

Die etwa 75 Kunden von innominate zeigen, daß sich die ganze Branchenvielfalt für Linux interessiert. Darunter befinden sich die Karl Blatz Gruppe, Produzent der Kinderfernsehsendungen "Bibi Bloxberg" und "Benjamin Blümchen," das Deutsche Institut für Normung (DIN), das Deutsche Institut für Bautechnik, der Internet-Serviceprovider X-Online und die Bundesdruckerei.

Unter den Motiven, sich auf Open Source-Software zu konzentrieren, nennt Kerstin Tober, freiberufliche Beraterin bei innominate, das Preis-Leistungs-Verhältnis, die Offenheit und die Stabilität sowie die wachsenden Akzeptanz für den Linux-Markt. Sie preist das schnelle und hochwertige *Feedback* der Linux-*Community* auf Probleme. Innominate selbst hat Bug-Reports und Feature-Vorschläge gemacht und einige kleine Patches beigesteuert.

---

[529] Kerstin Tober, Wizards 7/1999

Zu den Nachteilen zählt Tober die Haftungsunsicherheit, den Mangel an komplexen Applikationen, bestimmte technologische Schwächen gegenüber kommerziellen Unixen bei *High-End*-Anforderungen wie *High-Availability* und Schwächen in der Nutzerfreundlichkeit gegenüber MS-Windows. Bei allen Vorteilen der freien Software sieht Innominate auch weiterhin einen Platz für proprietäre Software. “Solche Projekte wie Lotus-Notes oder andere komplexe Anwendungen, in dem wirklich Mann-Jahre an Software-Entwicklungsarbeit stecken, für die wird es sich nicht unbedingt rechnen, die Sourcen frei zugeben. ... Es soll kein Dogma werden, daß alle plötzlich ihre Quellen freigeben müssen.”[530]

New Technologies Management GmbH

Rudolf Strobl ist seit Anfang der neunziger Jahre mit diversen Firmen, angefangen bei ARTICON, im Linux-Bereich aktiv. 1994 gründete er das Linux-Magazin.[531] 1996 verließ er ARTICON, gründetet Cybernet und darunter die Firma Linuxland. Ende 1998 verließ er Cybernet und gründete die Firma New Technologies Management GmbH (NTM), die Linux-Firmen dabei unterstützt, ihren Markt zu finden.

Firmen, die in der ersten Hälfte der neunziger Jahre, oft von Studenten, gegründet wurden, bauten auf Idealismus und eine gute Idee. “Früher hat man klein angefangen, die Zahl der User ist gestiegen, und man konnte mit dem Markt mitwachsen. Das ist heute ein Problem. Die technische und betriebswirtschaftliche Kompetenz konnte ebenfalls mit der Firma wachsen. Wenn man heute wirklich auf die Beine kommen will, sollte man eigentlich schon vorher wissen, was ein Businessplan ist. Man sollte betriebswirtsschaftliche Auswertungen lesen können, sollte ein bißchen was wissen zum Marketing, weil einfach die Konkurrenz stärker geworden ist.”[532]

Früher waren die Manager dieser Firmen selber alle Anfänger. Heute ist das nicht mehr so, weil es eben schon arrivierte Firmen gibt. Außerdem migrieren Firmen aus dem Windows-Bereich, wo die Konkurrenz immer schon da war, zu Linux. Ein Start bei Null ist somit schwieriger.

“Erfolgsfaktoren waren früher technisches Know-how und viel Enthusiasmus. Heute würde ich sagen, sind es gute Managementfähigkeiten. Und Ideen und Kapital werden immer wichtiger. Das ist nicht zuletzt zu sehen an den Gründungen oder Beteiligungen durch VC's [Wagniskapitalinvestoren] bei Innominate und ähnlichen Firmen. Früher finanzierte sich das Firmenwachstum aus dem Umsatz. Die Geschwindigkeit wäre heute zu gering, um ein *major Player* zu werden. Kapital ist auf alle Fälle ein Thema.”[533]

Die Finanzierung einer Neugründung ist eine Herausforderung. Ideal ist natürlich eine Eigenfinanzierung, aber die ist nicht immer möglich. Dann gibt es die klassischen Instrumente: Bankenfinanzierung und Finanzierung durch Fördermittel. Doch Banken wollen einen Businessplan sehen und wissen, wie sich der Linux-Markt entwickelt. Sie würden gern den Geschäftsmechanismus verstehen: ‘Wie verdient man denn da überhaupt Geld? Was verkaufen Sie denn?’ Leuten, die Handfestes gewohnt sind, ist das nicht immer einfach zu erklären. So bekommt man deutlich einfacher eine Finanzierung in Millionenhöhe für ein leerstehendes Bürogebäude, weil es dann eben das Gebäude und ein Grundstück gibt, als für

---

[530] Tober, Wizards 7/1999

[531] http://www.linux-magazin.de

[532] Rudi Strobl, Wizards 7/1999

[533] Strobl, Wizards 7/1999

eine zündende Geschäftsidee. Deutsche Banken agieren nach Einschätzung Strobls hier noch deutlich konservativer als amerikanische.

Eine Finanzierung durch Fördermittel sieht zunächst sehr attraktiv aus. Allerdings ist hier eine Vorgabe, daß man mit dem Projekt noch nicht angefangen haben darf, wenn man den Antrag stellt. In einem Markt, der so dynamisch ist wie bei Linux, ist das ein Problem. Wenn man sechs bis acht Monate wartet, bis über den Antrag entschieden wird, kommt man mit seiner Idee meistens bereits zu spät. Zudem werden auch Fördermittel über eine Bank ausgezahlt, die sicherstellen muß, daß das Geld wieder zurückfließt. Die kapitalsuchende *Startup*-Firma steht also wieder vor denselben Schwierigkeiten, wie bei einer direkten Bankenfinanzierug.

Im Endeffekt, so Strobl, ist die erfolgversprechenste Methode eine Finanzierung über *Venture Capital*. *Venture*-Kapitalisten sind spezialisiert auf die Probleme bei Firmengründungen, z.B. daß die erwarteten Umsätze nicht eintreffen und Kapital nachgeschossen werden muß. Desweiteren sind VCs typischerweise auf *Hightech*-Unternehmen, z.B. Internet-Dienstleister wie Yahoo, spezialisiert und mit den Bedingungen des neuen Marktes vertraut. Schließlich bringen sie solche Firmen an die Börse, weil sie ihr Geld zurückbekommen wollen.

Die New Technologies Management GmbH will nun Linux-*Startups* auf die Beine helfen. NTM übernimmt z.T. Kapitalbeteiligungen in einem *Startup*. Ein solches *Lead-Investment* kann größere *Follow-up*-Investitionen von Dritten nach sich ziehen, da ein VC viel eher in einen *Start-up* geht, in den schon jemand investiert hat, noch dazu jemand, der die Szene kennt. Vor allem aber vermittelt und koordiniert NTM passenden VCs. Bei den VCs gibt es extrem große Unterschiede. Neben denen, die die Geduld für diesen neuen Markt aufbringen gibt es solche, die möglichst rasch die Mehrheit einer Firma bekommen wollen. Das bedeutet häufig, daß die Gründer aus den Managementpositionen gedrängt werden, wenn der VC nervös wird, weil die Umsätze nicht schnell genug eintreffen oder die Richtung nicht zu stimmen scheint. Oft geht durch solche erzwungenen Umbesetzungen an der Spitze die Kreativität der Firma, das geistige und kreative Potential verloren.

NTM fungiert außerdem als Berater und unterstützt die Gründer beim Erstellen eines Businessplans als Basis für alle weiteren Aktivitäten, da nach Strobls Erfahrung die wenigsten Firmengründer einen Businessplan erstellen können, der von einer Bank oder einem VC akzeptiert würde. NTM kann auch beim Aufbau von Management- und Firmenstruktur, beim *Workflow*, bei der Organisation und beim Aufbau des Vertriebs helfen und Dienstleistungen wie Buchhaltung, *Financial Controlling* und juristische Unterstützung übernehmen. Nicht zuletzt verspricht NTM Synergieeffekte mit anderen Firmen im Verbund, zu dem auch die unter LunetIX genannte Linux Information Systems AG gehört, und mit Firmen, zu denen partnerschaftliche Beziehungen bestehen.

NTM will, im Gegensatz zu herkömmlichen VCs, ein Teil der Linux-*Community* bleiben. “Wir wollen ein *Enabler* und Moderator für *Business Opportunities* sein, um die Firmen zu unterstützen.”[534]

die tageszeitung

Die taz setzt offene Software seit mehr als zehn Jahren ein, als erstes im Bereich Mail-, News- und Domain Name-Server und mit Software wie Bind und Sendmail sowie den

[534] Strobl, Wizards 7/1999

gesamten GNU-Tools: dem GCC und den weiteren Entwicklungs-Tools. Ein größeres Projekt war es, das gesamte interne Volltextarchiv auf WAIS (Wide Area Information Servers) umzustellen. Es folgten Web-Server innerhalb des taz-Intranets und, als externe Web-Server, zuerst CERN HTTPD und dann Apache.

Die taz-EDV lief anfangs unter Sun-OS und wurde dann nach Solaris-2 portiert. Seit etwas 1995 wird Linux eingesetzt. Die Systemadministratoren und Software-Entwickler Ralf Klever und Norbert Thies begannen mit einem kleinen System von Slackware, das sie nach und nach bis zum heutigen Stand (Juli 1999) ausbauten. Den Anstoß gab, daß Linux zu der Zeit erstmals das X-11-Objekt-Format unterstützte, dessen Vorteile sie bereits vom Sun-OS kannten.

Die Einsatzgebiete von Linux heute sind Mail- und Web-Server, Firewall, Archivsystem und das Redaktionssystem. Den Anfang machten 1995 ISDN-Router und Web-Server unter Linux. Zur der Zeit begann man, im Internet vorsichtiger zu werden und zwischen das eigene Firmennetz und das Internet eine Schutzmauer, eine Firewall zu errichten. Den Ausschlag für Linux gab die Tatsache, daß die Firewall-Software im Quelltext verfügbar ist. Anders als bei einem anonymen, proprietären Produkt wie die Firewall-1 von Sun, konnten die taz-Entwickler hier den Quelltext überprüfen und genau feststellen, wie die Filter funktionieren.

Das Archivsystem unter Linux für Texte und Fotos haben Thies und Klever ursprünglich auf Sun entwickelt. Heute laufen pro Tag 2.000 Agenturmeldungen ein, die direkt indiziert und in die Redaktionen verteilt werden. Dieses Archiv umfaßt einen Zeitraum von drei Jahren. Das Textarchiv der taz selbst beinhaltet die Volltexte von mehr als 12 Jahren, da die Zeitung nahezu seit ihrer Gründung die Texte elektronisch archiviert hat. Auch die Fotodatenbank läuft unter Linux, hinter dem WAIS als Indizierer dient. Hier laufen 500 Bilder pro Tag ein, plus weitere Agenturfotos und Bilder von den eigenen Fotografen. Klever betont, daß das System zuverlässig und rund um die Uhr laufe. Es müsse nur alle halbe Jahre um einige Gigabyte an Festplatten erweitert werden.

Das Redaktionssystem unter Linux ist ebenfalls eine Eigenentwicklung der taz. Es besteht aus einem Editor zur Eingabe der Texte, der gesamten Ablaufsteuerung, Zugriffskontrollen, Anzeigen über den Produktionsstatus, Textdatenbanken usw. Diese Funktionen sind als Intranet-Lösung realisiert worden und laufen als Module von Apache und als Dämonen, die im Hintergrund die Texte verwalten. Zunächst wurden bei der taz die Entwickler- und Administrationsarbeitsplätze auf Linux umgestellt und mit der Einführung des Redaktionssystems schließlich alle Arbeitsplätze innerhalb der Redaktionen. Heute sind etwa 140 Linux-Workstations im Einsatz

Auf den Redaktions-Workstations ist das einzige kommerzielle Produkt, die Office-Software Applixware, im Einsatz, so daß sich die Software-Kosten für einen Arbeitsplatz auf etwa 70 DM belaufen. Auch als Layout-System wird das proprietäre Programm Framemaker verwendet, aber Klever denkt, daß möglicherweise K-Word eine Alternative darstellen könnte.

Die Softwarekosten pro Server belaufen sich auf exakt Null Mark. Alle Programme stammen aus dem Internet und werden im Haus angepaßt und erweitert. Linux ist schließlich auch attraktiv, weil es auf preiswerter Hardware läuft. Als wichtigsten Grund für freie Software nennt Klever jedoch ihre Flexibilität. Beispielsweise wollten sie einen Radius-Server außerhalb ihrer Firewall einsetzen, allerdings ohne dort Paßwörter vorzuhalten. Da ihnen der Quellcode zur Verfügung stand, konnten sie sich ansehen, wie die Authentifizierung arbeitet und sie entsprechend ändern.

Die Standzeiten von Linux im Server-Bereich entsprechen nach Einschätzung der beiden taz-Entwickler denen von Sun-OS oder Solaris. Firewall und Web-Server laufen 3-4 Monate ohne Re-Boot durch. Auch die Desktops seien wartungsarm. Mit Hilfe von Shell-Scripts werden Updates einfach auf die einzelnen Workstations gespielt.

Neben der sehr guten Verfügbarkeit und Testmöglichkeit hebt Klever die Qualtität und den guten Support für freie Software hervor. Durch ihre weite Verbreitung im Internet werden Fehler schnell bekannt und behoben. In den Newsgroups oder direkt von den Entwicklern bekomme man sehr schnell kompetente Antworten auf seine Fragen. “Das sind alles Aspekte, die uns proprietäre Betriebssysteme und Software in dieser Form nie hätten geben können.”[535]

Babcock-BSH

Das Unternehmen Babcock-BSH befaßt sich mit der Planung, Konstruktion und Lieferung von Anlagen, Systemen, Apparaten und Komponenten der Verfahrenstechnik. Schwerpunkt ist dabei die thermische Behandlung von Schüttgütern aller Art in den verschiedensten Industriezweigen: in der chemischen Industrie, Nahrungsmittelindustrie und der Pharmaindustrie.

Bernd Driesen, Leiter der Datenverarbeitung bei Babcock, sieht Linux als ein ideales System, um die verschiedenen Rechnerwelten des Unternehmens miteinander zu verbinden. Es sei äußerst kostengünstig, und es gebe, anders als oft behauptet, sehr guten Support, sowohl über die Newsgroups, als auch über die Support-Datenbank ihres Providers.

Linux wird als Bindeglied zwischen der Unix-Welt (AIX) und der von MS-Windows und Lotus-Notes eingesetzt. Babcock hat zwei vollkommen unterschiedliche Rechnerwelten im Einsatz. Das ist zum einen ein Tokenring-Netzwerk mit RS/6000-Rechnern von IBM unter AIX, die als CAD-Workstations mit der Software CC-Draft eingesetzt werden. Zum anderen gibt es PC-Arbeitsplätze mit Windows 95 in einem Ethernet. Hier kommt Novell-Netware als Server zum Einsatz. Auf den PCs laufen diverse Anwendungen, in der Hauptsache jedoch leider, wie Driesen sagt, Office 97. Für die Kommunikation innerhalb des Unternehmens wird auch die Groupware Lotus-Notes eingesetzt. Früher mußten deswegen PC-*User* die CAD-Zeichnungen manuell per FTP von den Unix-Rechnern herunterladen, um sie dann von den PCs aus verschicken zu können, eine fehlerträchtige Vorgehensweise. Darum wurde Linux in einem ersten Schritt eingesetzt, um die CAD-Daten per NFS zu *mounten*, um sie dann per Samba der PC-Welt zur Verfügung zu stellen. Die Anwender konnten jetzt ihre Dateien einfach von einem Laufwerk auf ein anderes schieben, und der Dateitransfer war vollzogen. Weil dabei eine zusätzliche Installation auf den Clients erforderlich war, hat Driesen später den Samba-Server durch den Netware-Emulator Mars ersetzt. Der Mars-Emulator gibt die Unix-Laufwerke als ganz normale *Netware-Volumes* frei. Da alle PCs grundsätzlich mit der Client-Software für den Zugriff auf einen Novell-Server ausgestattet sind, braucht der Administrator nur noch den Benutzer in eine bestimmte Gruppe aufzunehmen, ein Vorgang, der nur noch fünf Sekunden dauert. Startschwierigkeiten, z.B. mit der Übertragungsgeschwindigkeit, konnten sehr schnell mit Hilfe der News-Groups und der Support-Datenbank des Distributors gelöst werden.

Die 120 Windows-PCs werden mit Hilfe einer einzigen Linux Boot-Diskette installiert. Darauf befindet sich ein komplettes Linux mit allen wichtigen Funktionalitäten.

---

[535] Klever und Thies, Fachgespräch 7/1999 & Wizards 7/1999

Dadurch, daß Linux beim Booten automatisch die richtige Netzwerkkarte (inklusive I/O-Adresse und IRQ) findet, hat der Rechner nach dem Booten von der Diskette einen direkten Zugriff auf den Support-Server. Von dort wird dann mit Hilfe von tar eine vorbereitete Windows-Installation herunterladen. Innerhalb von etwa einer halben Stunde läßt sich so ein komplettes laufendes Windows-System, mit allen Standardprogrammen, den wichtigsten Netzwerkdruckern, Internet-Browser usw. installieren.

Ein Intranet-Webserver hilft dem Anwender, quasi als Menüsystem, schnell auf die verschiedenen Dienste, die für die Firma wichtig sind, zugreifen zu können. Hierfür wird der Apache genutzt, dessen *out of the box*-Installation den Anforderungen genügt. Die schwierige Administration des Apache sei daher, so Driesen, erst einmal kein Thema, höchstens eine Herausforderung.

Linux übernimmt außerdem als *Router* die Anbindung des lokalen Netzes an das Internet. Nachdem 1997 die Fertigung ge-outcourced wurde, müssen täglich Daten, wie Office-Dokumente, aber hauptsächlich CAD-Zeichnungen, zu den externen Fertigern geschickt werden. Die Standard-Firewall von Linux und als Proxy-Server der Squid kommen ebenfalls zum Einsatz. “Und gerade in dem Bereich gibt uns das Open Source-Projekt Sicherheit. Wir könnten uns, wenn wir Zeit hätten, den Quellcode anschauen, um da eventuell noch Sicherheitslücken zu finden. Machen wir natürlich nicht, aber wir haben zumindest das Gefühl der Sicherheit.” Die Homepage der Firma läuft aus Sicherheitsgründen und wegen der hohen Standleitungspreise auf einem Server des Providers.

Nachteile von Linux hat Driesen in all diesen Einsatzgebieten noch keine festgestellt. Abstürze habe es bisher keine gegeben. Die meisten Dienste laufen monatelang absolut störungsfrei.

Lehmanns Buchhandlung

Lehmanns ist eine Kette von Fachbuchhandlungen mit dem Schwerpunkt auf Informatik, Medizin, Naturwissenschaften, Technik und Wirtschaft und mit Filialen in 20 Städten in Deutschland. Lehmanns begann bereits 1992 mit eCommerce, und das auf Linux. Als erste Buchhandlung weltweit hat es mit dem Lehmanns Online Bookshop (LOB)[536] Bücher über das Internet vertrieben.

Die beste Software taugt ohne Handbücher nur die Hälfte. Bernd Sommerfeld, Buchhändler bei Lehmanns Berlin, übernahm 1985 die junge EDV-Abteilung und baute sie zur “UNIX-Buchhandlung in Deutschland” auf. Ab 1990 waren hier auch schon die Bücher der Free Software Foundation zu GNU Emacs, Lisp oder gcc erhältlich, ebenso Software wie Minix und Coherent (das erste kommerzielle PC-Unix). 1992 hörte Sommerfeld zum ersten Mal durch Studenten der FU-Berlin und durch Diskussionen auf der Usenet-Newsgroup comp.os.minix von Linux.

> “Linux gab es zur dieser Zeit ausschließlich im Internet per ftp. Ich aber wollte es auf Disketten an alle die verkaufen, die den Zugang zum Netz noch nicht hatten. Sebastian Hetze, damals Mathematik-Student in Berlin, war gerne bereit, eine Distribution ‘Lunetix lsd’ auf sieben Disketten mitsamt photokopiertem Handbuch zu erstellen. Ich erinnere mich gut, wie zur Cebit 1993 der Messestand von Lehmanns wegen des ersten Linux- Anwenderhandbuchs und der ersten Distribution auf

[536] http://www.lob.de/

> Disketten regelrecht gestürmt wurde. Auch Verleger und Softwarefirmen witterten interessante Geschäfte."[537]

Aus sieben Disketten wurden schnell über hundert, und man ging zum Pressen von CD-ROMs über. 1994 publizierten aufgeschlossene Computerbuch-Verlage die ersten Linux-Bücher, darunter das bereits genannte Linux-Anwenderhandbuch von LunetIX. Neue Distributoren wie Suse, Caldera, Yggdrasil, DLD und LST brachten Linux-CDs heraus. Linux-Zeitschriften, das amerikanische Linux Journal und das deutsche Linux-Magazin, kamen auf den Markt. Sommerfeld gibt selbst verschiedene Linux-Distributionen heraus. Auch durch die Initiierung des ersten internationalen Kongresses "Linux & Internet" 1994 in Heidelberg machte er sich um Linux verdient. 340 Entwickler und Freunde freier Software aus Finnland, England, den USA und Frankreich trafen sich dort zum ersten mal *live*.

Bei Lehmanns selbst ist Linux seit Februar 1993 als Mail- und bald darauf als Webserver im Einsatz. Die eCommerce-Lösung für den Lehmanns Online Bookshop wurde von LunetIX (s.o.) entwickelt. LOB bietet eine komfortable Suche im Kataloge der lieferbaren Bücher, CDs und Videos. Z.T. gibt es durchsuchbaren Abstracts. Die Bestellungen erfolgt über einen Warnkorb und wird per eMail bestätigt. In einem Vergleichstest verschiedener Online-Buchläden durch die c't ließ LOB in der Gesamtwertung und vor allem bei den Suchfunktionen, der Übersichtlichkeit, Handhabung und Lieferzeit fast alle Konkurrenten hinter sich.[538]

Sommerfeld lobt vor allem die Stabilität des Linux-Systems. Der Rechner läuft zwei Jahre lang weitgehend wartungsfrei und bootet selbst nach Stromausfall während des laufenden Betriebs ohne größere Datenverluste. "Ich als Buchhändler hatte wenig Erfahrung mit Linux und habe es dem Linux überlassen, sich selber immer wieder hochzufahren. Wenn ich dagegen sehe, wie oft am Tag unsere Rechner, die mit Windows laufen, abstürzen, ist es für mich traumhaft, wie stabil so ein Betriebssystem ist."[539]

Individual Network

Ein Beispiel aus dem nichtkommerziellen Bereich ist das Individual Network (IN).[540] Das IN ging aus der Berliner Mailbox-Szene hervor, die 1980 den Austausch von Mail und News per UUCP kennen- und über die TU und die FU nutzen lernte. 1990 schlossen die Berliner Sites eine Vertrag mit UniDo über UUCP-Zugang, reservierten die Domain IN-BERLIN.de und gründeten das IN-Berlin.[541] Ein Jahr darauf entstand zusammen mit drei ähnlichen Gruppen das das bundesweite Indiviual Network. Der Individual Network e.V. vertritt als Solidargemeinschaft das Ziel eines kostengünstigen privaten Zugangs zum Internet. Es tritt als als Einkaufsgemeinschaft auf, die von EUnet, DFN und anderen Netzbetreibern IP-Kontigente erwirbt, um sie seinen Mitgliedern für private, nichtkommerzielle Nutzung zur Verfügung zu stellen. In einer Zeit, als Internetzugang außerhalb der Universitäten kaum verfügbar und wenn, sehr teuer war, stellte der IN für viele nicht nur die einzige Möglichkeit dar, Internetzugang zu bekommen, sondern auch einen Ort, um Gleichgesinnte zu treffen und

---

[537] Sommerfeld 1999: Kap. 7

[538] Kuri 19/1999

[539] Sommerfeld, Fachgespräch 7/1999

[540] http://www.individual.net/

[541] http://www.in-berlin.de

mehr über die Technologie zu lernen. Nicht wenige kommerzielle ISPs, wie z.B. das Berliner SNAFU, wurden von ehemaligen Mitgliedern des IN gegründet. 1996 hatte diese 'Graswurzelbewegung' der Netzkultur IP-Zugängen in 84 Städten Deutschlands. Es gab knapp 60 sog. Regional-Domains mit knapp 7.800 Rechnern und schätzungsweise 70.000 Teilnehmern. Damit dürfte das IN der größte reine Internetprovider in Deutschland gewesen sein.[542]

In der Frühzeit des IN wurden NeXT- und Sun-Rechner oder K9Q-Router eingesetzt. Das WWW war noch nicht erfunden. Das Internet außerhalb der kostspieligen Standleitungsinfrastruktur beruhte auf UUCP und bot im wesentlichen die Dienste Mail und News. Im Laufe der Zeit wurden beim IN-Berlin alle Rechner auf Linux umgestellt und einer auf BSD. "Wir können uns über mangelnde Qualität der Software wirklich nicht beklagen. Wir haben unter unseren aktiven Leuten auch einige Entwickler, die auch an Linux-Projekten mitarbeiten, bsw. jemand, der hauptberuflich bei AVM arbeitet und auch ISDN-Treiber für Linux entwickelt. Von daher können wir, wenn es Probleme gibt, direkt eingreifen."[543]

O'Reilly Verlag

O'Reilly & Associates[544] ist 1978 als Consulting-Firma für technische Schriften entstanden, die z.B. Unix-Handbücher verfasste, die Workstation-Anbieter zusammen mit ihrer Hard- und Software ausliefern. Mitte der Achtziger begann sie, die Rechte an ihren Auftragswerken zu behalten, um sie auch an andere Software-Herstellern zu lizenzieren. Spätestens auf der MIT X-Konferenz 1988 wurde deutlich, daß es einen großen Bedarf nach Handbüchern gab, die unabhängig von der Software zu erwerben sind. Aus den Auftrags-Dokumentationen wurden Bücher und aus O'Reilly ein Verlag. Heute hat O'Reilly 120 Titel im Programm mit Schwerpunkten auf Unix, X, Internet und anderen offenen Systemen. Vor allem unter Entwicklern hoch angesehen sind Referenzwerke zu freier Software (Apache, Debian GNU/Linux, Linux-Kernel, GNU Emacs und andere GNU-Software, TCP/IP, Python, sendmail, Samba). Der Verlag beschäftigt über 200 Menschen und hat Niederlassungen in den USA, Japan, Frankreich, Deutschland, Großbritannien Taiwan und der Volksrepublik China.

An der großen Nähe zur technischen Community merkt man, daß O'Reilly aus der Computer-Welt und nicht aus dem traditionellen Verlagsgeschäft kommt. Alle Lektoren sind ehemalige Programmierer. Die Auflagen sind bewußt klein, um in häufigen Nachdrucken die Leserreaktionen und die rasche Änderungen der Software aufnehmen zu können.

O'Reilly hat sich früh auch mit Online-Publishing beschäftigt. Software-Hersteller wollten ihre Dokumentation auch online anbieten. In den Achtzigern entstanden mehrere Dutzend Formate für die digitale Präsentation von Handbüchern und Hilfetexten, z.B. AnswerBook von Sun oder InfoExplorer von IBM. Der O'Reilly-Verlag hätte seine Bücher in zahlreichen unterschiedlichen Formaten anbieten müssen. Um der Abhängigkeit von den verschiedenen Formaten zu entkommen, begannen er zusammen mit HaL Computer Systems

---

[542] Ingmar Camphausen, Internetzugang via Individual Network Berlin e.V., 1997, http://www.in-berlin.de/public/representation/vorstellung.shtml. S.a.Infos ueber den Individual Network e. V., http://www.individual.net/deutsch/verein/ziele.shtml

[543] Ronneburg, Fachgespräch 7/1999

[544] http://www.ora.com

1991 eine SGML-DTD (Document Type Definition) namens DocBook[545] für Online-Handbücher zu entwickeln. Der Kreis, der diesen freien Standard entwickelt, erweiterte sich zur Davenport Group. DocBook hat sich unter den offenen Systemen und auch in der Linux-Community weithin durchgesetzt.

1993 entdeckte Tim O'Reilly das Web und damit HTML, einen Dialekt von SGML. Aus Arbeiten an einem Browser für das DocBook-Format und aus Viola, eine Art HyperCard-Clone unter Unix, ein Tool-Kit zum Bauen von Hypertext-Applikationen, entstand der Global Network Navigator (GNN). Sein Inhalt bestand anfangs aus dem Internet-Verzeichnis der ersten 300 Websites aus dem von O'Reilly herausgegebenen "The Whole Internet User's Guide & Catalog" von Ed Krol. GNN begann als Demo und wuchs sich zu einem neuen Informationsdienst mit Nachrichten, Artikeln und Rezensionen über Internet-Dienste aus, eine Art Informations-Interface als Pforte zu den eigentlichen Diensten. GNN wurde 1995 an AOL verkauft. Mit "Web Review" und dem "World Wide Web Journal" setzt der Verlag seine Online-Publishing-Aktivitäten fort. Im Bereich Online-Bücher werden heute thematische Auswahlen aus dem Verlagsprogramm zu Handbibliotheken zusammengestellt. Verfügbar sind derzeit Referenzwerke zu Java und zu den Schlüsseltechnologien für Webmaster. Verfügbar sind sie im Web, in HTML mit Querverweisen und Suchmaschine, zum Preis von $ 59,95 im Jahr.

Auch in der Software-Entwicklung hat sich O'Reilly engagiert. Wer in den Anfangsjahren Internetzugang haben wollte, mußte sich die nötige Software zusammensuchen. Gemeinsam mit der Software-Firma Spry enwickelte O'Reilly das "Internet in a Box", das Spry-Software, GNN und wiederum den Whole Internet User's Guide and Catalog enthielt. Es folgte WebSite Professional für Webseitenentwickler, ein Webserver für Windows NT and Windows 95 und weitere Produkte.

Der O'Reilly Verlag hat nicht nur Handbücher für freie Software wie Bind und Perl verlegt, sondern unterstützt sie auch durch Veranstaltungen, wie dem Open Source Summit im April 1998 und die jährlichen Perl-Konferenzen, Software wie den Perl Resource Kit und durch das Hosting von perl.com. Er ist dabei immer darum bemüht, freie Software auch im kommerziellen Software-Markt zu stärken.[546]

Tim O'Reilly schätzt, daß er im Laufe der Jahr Bücher über Open Source-Software im Wert von mehr als 100 Mio Dollar verkauft hat und sein Open Source-bezogener Umsatz im vergangenen Jahr größer war, als der von SuSE oder Red Hat. Dennoch ist er der Ansicht, daß nicht er, sondern Bill Gates der größte Open Source-Profiteur sei, da er offene Standards, Protokolle und Software in seine proprietäre Software integriert und seinen Kunden Upgrades für MS-Windows und Office verkauft, damit sie Internet-Funktionalität erhalten.[547]

Intershop

Der heutige Marktführer für eCommerce-Lösungen wurden 1992 in Jena gegründet. Intershop[548] zog relativ schnell *Venture Capital* an und verlegte seinen Hauptsitz ins Silicon

---

[545] 1998 löste sich die Davenport Group auf und die Maintainer-Rolle für DocBook ging an OASIS über, http://www.oasis-open.org/docbook/

[546] Tim O'Reilly, History & Company Overview from President, 1998, http://ora.com/oreilly/tim.html

[547] Tim O'Reilly, Wizards 7/1999

[548] http://www.intershop.de

Valley, aber beläßt seinen Entwicklungsstandort mit 200 Software-Ingenieuren in Jena. Heute hat das Unternehmen 14 Büros rund um die Welt. Im Juli 1998 ging es an die Börse mit einer Marktkapitalisierung von heute ungefähr drei Milliarden DM. Für 1999 rechnete Intershop mit einem Umsatz von 50 Millionen Dollar. Ende das Jahres sollte es erstmals den *Break Even Point* erreichen. Vision des Unternehmens ist es, 17-20% des eCommerce-Marktes zu halten.

20.000 Unternehmen nutzen Intershop-Software, um Produkte im Internet anzubieten. Das Produkt bietet eine *Storefront* mit der gesamten Warenwirtschaft dahinter, Integration in *Enterprise Resource Planning*-Systeme, *Customer-Relation-Managementsysteme* und was noch zu einem komplexen eCommerce System gehört. Die Software bewegt sich im höheren Preisbereich. Die Kosten fangen bei minimal 5.000 Dollar an und gehen dann bis zu einer Million Dollar oder mehr.

Intershops Kunden sind Telekommunikationsunternehmen, die für ihre Kunden eShops hosten. Eine zweite Kundengruppe sind die *Enterprises*, mittlere bis größere Unternehmen, die mit eigener IT-Infrastruktur eCommerce betreiben. Zu den Kunden gehören auch Systemintegratoren wie KPMG oder Price-Waterhouse und kleinere Design-Agenturen, die für ihre Kunden wiederum Läden im Internet eröffnen.

Intershops Haltung zu Open Source-Software ist zurückhaltend und unenthusiastisch. Die Produktpalette von Intershop wird seit April 1999 auch unter Linux angeboten, wohl vor allem, um die Aufmerksamkeitswelle auzunutzen. Die Verkaufszahlen der Linux-Versionen sind allerdings bislang gering. In den IT-Systemen von Intershops Kunden werden hauptsächlich NT und Solaris eingesetzt, und es gebe kein Interesse, dies zu ändern. Frank Gessner, Mitbegründer und *Vice President Engineering* von Intershop, erwartet nicht, daß der Linux-Hype in den nächsten 12 Monaten nennenswerte Folgen für ihren Markt haben wird. Er rechnet auch nicht damit, daß sich die Lizenzpreise für Software allgemein in den nächsten Jahren drastisch reduzieren werden.

Gleichzeitig betont Gessner die Vorteile, die Intershop davon hat, viele Informationen und Ideen aus dem Internet beziehen zu können. Das Unternehmen setzt vor allem Java ein und gibt eigenes Wissen an die Java- und an andere Newsgroups zurück.

Die Intershop-Software selbst ist nicht Open Source. “Wir haben gelernt, daß man Leuten, die sich selbst helfen können, sich selbst helfen lassen muß,”[549] sagt Gessner, schränkt jedoch ein, daß das nicht einfach ist, wenn ein Produkt eine gewisse Komplexität überschritten hat. Er unterscheidet zwischen Produkt und Plattform: “Ich bin kein Jünger von Open Source, aber ein Jünger von offenen Standards.” Wer ein Produkt kaufe, möchte auch von seiner Weiterentwicklung profitieren, aber nicht selbst in den Source-Code eingreifen. Die Entscheidung gegen eine Offenlegung des Codes begründe sich nicht mit dem Schutz des ‘geistigen Eigentum’, sondern mit der Unterstellung, die Kunden wollten ohnehin nicht in die über eine Millionen Zeilen Quelltext der Intershop-Software eingreifen.

Im Quelltext verfügbar sind allein die APIs und die Dokumentationen (java.doc, db.doc usw.). In Ausnahmefällen wird anscheinend “trainierten Systemintegratoren” der Source Code verfügbar gemacht. “Die wissen auch, wenn sie das machen, daß sie oft alleine damit zurechtkommen müssen. Das ist ja bei Open Source der Vor- und Nachteil in einem.”

Auf die Frage nach der *Community* sagt Gessern, das sei ganz einfach: Man müsse die Leute zum Kommerz befähigen, ihnen Wege aufzeigen, wie sie, wenn sie um eine Technologie herum investieren, ein *Return of Investment* erhalten. Sprachs und forderte die

[549] Gessner, Wizards 7/1999

versammelten Open Source-Entwickler auf, sich von Intershop einstellen zu lassen. Gessner: “Wir sind eine Firma, die nicht davor zurückschreckt, Geld zu verdienen.”

Sicherheit

Der Problemkomplex Sicherheit läßt sich in die Bereiche Betriebssicherheit (Stabilität, Verfügbarkeit, Investitionssicherheit, Haftung) und Systemsicherheit (Abhörschutz, Kryptografie, Firewalls) gliedern. Die Experten vom Bundesamt für die Sicherheit in Informationstechnik (BSI) über Bundeswirtschafts- und Forschungministerium bis zum Chaos Computer Club (CCC) sind der einhelligen Überzeugung, daß die Quelloffenheit einer Software essentielle Voraussetzung für ihre Sicherheit unter allen Gesichtspunkten ist. Proprietäre Software, bei der der Anwender sich einzig auf das Wort des Anbieters verlassen kann, muß grundsätzlich suspekt sein. Überprüfbarkeit durch Experten des eigenen Vertrauens, d.h. Quelloffenheit, ist Vorraussetzung für Vertrauensbildung. Besonders die Vertreter des BSI sagten mit deutlichen Worten, daß sie lieber heute als morgen auf MS-Windows-Software verzichten würden.

Dies steht im Kontrast zu einem erheblichen Mißtrauen in Bezug auf Sicherheitsfragen, auf das Open Source-Software weiterhin trifft. Ingo Ruhmann (BMBF) hält vertrauensbildende Maßnahmen und eine Öffentlickeitsarbeit in erster Linie durch die Distributoren für erforderlich, die breite Anwenderkreise über die Sicherheitsvorzüge freier Software aufklären.

Umgekehrt herrscht an vielen Orten weiterhin ein unbegründbarer Vertrauensvorschuß gegenüber proprietärer Software. Unbegründbar, weil die zahlreichen bekanntgewordenen Mängel und sicherheitsbedenklichen Features eigentlich jedes Vertrauen in proprietäre Software zunichte gemacht haben sollten. Zu der langen Kette gehört die mangelhafte Implementierung des Point-to-point-Tunneling-Protokolls von Microsoft,[550] die Fernsteuerung fremder Rechner mit Hilfe von MS Back-Orifice[551] oder die Identifizierungsnummern, die Intel in seinem neuesten Prozessor einbrannte, und die auch Microsoft nicht nur jeder einzelnen Version von Windows98 und Anwender*programmen* mitgibt, sondern die sich auch in Word-, Excel- und PowerPoint-*Dokumenten* findet. “Wenn man das in einem größeren Zusammenhang sieht -- für mich ist proprietäre Software etwas ganz Suspektes. Sie können das in eine Reihe setzen mit dem, was amerikanische Sicherheitsbehörden machen: auf der ganzen Welt eMails, Telefongespräche und Telefaxe aufzunehmen oder die Exportrestriktionen für Kryptoalgorithmen, dann ist für mich eine Software, in die ich nicht hineingucken kann, etwa genauso verdächtig.”[552]

Ebenso unbegründbar ist das Vertrauen in die stabilen Besitzverhältnisse der Unternehmen, auf die man sich in IT-Sicherheitsfragen verläßt.

> “1997 hat die Bundesregierung erklärt, daß im Bereich der Bundesbehörden 65.000 PCs mit Microsoft-Betriebssystemen im Einsatz sind. Heute werden es einige Tausend mehr sein. Keiner dieser PCs läßt sich auf Sicherheitsmängel überprüfen, ob er jetzt im Berlin-Bonn-Verkehr eingesetzt wird oder nicht. Im übrigen wurde die genutze Krypto-Hardware im Bonn-Berlin-Verkehr von einer Firma geliefert, die, nachdem der Kaufentschluß gefallen war, von der südafrikanischen Firma Persetel Q

---

[550] s. den Bericht darüber auf dem BSI-Kongreß 1999

[551] Die Hackergruppe “Cult of the Dead Cow” hat den Exploit des MS-Fernwartungswerkzeugs entdeckt, http://www.bo2k.com/. Schutz dagegen findet sich unter http://www.cultdeadcow.com/tools/bolinks3.html

[552] Rudolf E. Bahr, BSI, Fachgespräch 7/1999

> Holdings gekauft wurde, die während der Zeiten des Boykotts gegen Südafrika großgeworden sind, mit besonders guten Kontakten zum südafrikanischen Militär. Im Herbst 1999 wurde dieser Firmenteil immerhin von der deutschen Firma UtiMaco gekauft. Für Black Box-Systeme ist daran wichtig: Das Know-How um die Kyptomodule war für einige Zeit für Beteiligte offengelegt, deren Vertrauenswürdigkeit niemand geprüft hat. Hier gibt es einige Probleme, die dadurch nicht besser werden, wenn wir uns auf große kommerzielle Anbieter verlassen."[553]

Im *Office*-Bereich bedarf der Wechsel zu quelloffenen Systemen keiner weiteren Evaluierung mehr, trifft aber auf andere Schwierigkeiten. Zum einen ist Microsoft-Software in der Verwaltung und in der Wirtschaft heute Quasi-Standard. Eine Abteilung, die im Alleingang umsteigt, läuft somit Gefahr, nur in eingeschränktem Umfang Daten mit Kommunikationspartnern austauschen zu können. Zum anderen stehen unter Linux grafisch zu bedienende Oberflächen, die Anwender in diesem Bereich für ihre *Office*-Applikationen erwarten, erst seit relativ kurzer Zeit zur Verfügung. Die Anforderungen an den Anwender bei Installation und Wartung sind heute unter Linux noch deutlich höher als unter Mac-OS oder MS-Windows. Angesichts der kurzen Zeit, in der sich die grafischen Oberflächen KDE und Gnome und andere Merkmale der Benutzerfreundlichkeit entwickelt haben, ist jedoch damit zu rechnen, daß dieser Unterschied bald verschwinden wird.

Ruhmann fordert einen Wandel in der Sicherheitskultur. "Weg von dieser Aufregung: 'Der und der Browser hat mal wieder dies und jenes Sicherheitsloch', und der Panik, die dann hinterher kommt. Oder diese mittlerweile notorischen Meldungen über neue Viren, bei denen die Hälfte der Mails sich auf Viren bezieht, die gar nicht existieren. Diese Aufregung ist ja ganz schön und nützlich, sie führt aber letztendlich nur zu reiner Panikmache ohne vernünftige Differenzierung. Open Source-Software kann diese vernünftige Differenzierung dadurch leisten, daß es eine überprüfbare Sicherheit bietet, nur müssen dann auch die Mechanismen dafür her."[554]

Frank Rieger (CCC) sieht Mängel in der Sicherheitskultur auch auf Produzentenseite. Sicherheit ist häufig keine Priorität im Entwicklungsprozeß und kein strukturelles Element eines Produkts, sondern ein *add-on*, über das erst kurz vor der Markteinführung nachgedacht wird. Oft genug ist nicht klar, wer eigentlich dafür zuständig ist, daß ein Produkt am Ende sicher und vor der Auslieferung auch daraufhin getestet worden ist. Auch eine Zertifizierung bietet hier keine Gewähr, zumal, wenn sie wie im Falle von Windows NT zehn Jahren zuvor für ein komplett anderes Produkt errungen wurde. Schließlich werden Produkte, die von sich aus relativ sicher sein können, mit falschen Default- und Installationswerten ausgeliefert. Rieger führt als Beispiel eine aktuelle SuSE-Distribution, die nach einer Standardinstallation unnötig viele Ports offen hat. Auch wenn es möglich ist, ein SuSE-Linux wasserdicht zu machen, zeigt es sich doch bei allen Computer-Systemen, daß viele Endanwender die Default-Einstellungen nie verändern.[555]

Für Ruhmann bedeutet eine IT-Sicherheitskultur, "daß das Bewußtsein um IT-Sicherheit steigen muß; daß es kein Kostenfaktor ist, sondern ein erheblicher Nutzenfaktor; und daß der Grad der Panik aus Unwissenheit bei irgendwelchen Problemen dadurch herabgesetzt werden muß, daß die Leute auf der einen Seite einfach kompetenter mit IT

---

[553] Ingo Ruhmann, Wizards 7/1999

[554] Ingo Ruhmann, Wizards 7/1999

[555] Frank Rieger, Wizards 7/1999

umgehen können und auf der anderen Seite ein kompetenteres Beratungsangebot zur Verfügung haben. ... Qualität sollte nicht vom Glauben abhängig sein, sondern von überprüfbaren Kriterien."[556] Und Voraussetzung für eine Überprüfbarkeit ist Zugang zum Quellcode.

Betriebssicherheit

Auf der fundamentalsten Ebene ist nach der Betriebssicherheit von Systemen, nach ihrer Ausfallhäufigkeit, Stabilität und Verfügbarkeit zu fragen. In diesen Punkten sind Open Source-Systeme wie Linux durch eine weitergehende Fehlersicherheit proprietären Systemen deutlich überlegen. Freie Software ist natürlich nicht per se fehlerfreier, doch wenn ein Fehler auftritt, findet sich meist sehr schnell Unterstützung im Internet oder es kann ein Software-Entwickler angeheuert werden, der den Fehler beseitigt, während der Benutzer einer proprietären Software darauf angewiesen ist, daß der Hersteller den Fehler behebt.

Zur Investitionssicherheit ist oben unter "wirtschaftliche Potentiale" bereits einiges gesagt worden. Verantwortungsbewußte Firmen,[557] hinterlegen den Quelltext ihrer Software, so daß die Kunden weiterhin darauf zugreifen können, falls die Firma aufgekauft wird, bankrott geht oder aus anderen Gründen die Software nicht mehr warten kann.

Auch die Flexibilität und Anpaßbarkeit von quelloffener Software, die dort bereits genannt wurde, ist sicherheitsrelevant.

> "Wenn ich mir ein kleineres Sicherheitsproblem der letzten Wochen und Monate angucke, wird offenbar, daß Verbesserungen im Sicherheitszusammenhang immer wieder notwendig sind, und zwar im laufenden Betrieb. Die US-Navy hat festgestellt, daß Schiffe in der Adria im Kosovo-Krieg Mails senden und empfangen konnten und daß ihre Soldaten ziemlich blauäugig nicht nur Mails anderer Leute gelesen, sondern auch sensitive Informationen per Mail an ihnen unbekannte Leute weitergegeben haben. Was war die Lösung? Eingehende Mails wurden erlaubt, ausgehende nicht. Das funktioniert mit schlecht anpassbaren Systemen nur auf diese Art und Weise. Man hätte sich natürlich auch andere Wege überlegen können, diese Systeme ein wenig besser der neuen Situation anzupassen. Eine solche Anpassung ist natürlich bei *Black Box*-Systemen schlichtweg unmöglich. Bei Open Source-Systemen kann ich entsprechende Veränderungen jederzeit einbringen, wenn ich die Leute kenne oder selbst dazu in der Lage bin."[558]

Gewährleistung und Haftung

Ein immer wieder zu hörender Einwand gegen OSS lautet: 'Wie kann ich diesen "langhaarigen Leuten" trauen, die mir einen *Bug Fix* schicken?' Natürlich kennt man auch bei proprietärer Software in der Regel die Entwickler nicht persönlich. Der vermeintliche Unterschied besteht darin, daß man ihre Firma im Falle eines Fehlers verklagen könnte. "Hier wird also versucht, Sicherheit durch die Möglichkeit des Rechtsweges herzustellen."[559] Daß

---

[556] Ruhmann, Wizards 7/1999

[557] z.B. StarDivision, Ralf Hofmann, Fachgespräch 7/1999

[558] Ruhmann, Wizards 7/1999

[559] Ruhmann, Wizards 7/1999

diese Strategie bei kommerziellen Produkten auf theoretische und praktische Grenzen stößt, soll hier dargestellt und mit der Lage bei freier Software kontrastiert werden. Da freie Software niemandem gehört und keine großen Unternehmen die Gewährleistung dafür übernehmen, wie kann ein Nutzer sicher sein, daß die Software auch in kritischen Situationen das hält, was sie verspricht?

Zunächst ist anzumerken, daß auch kommerzielle Unternehmen für ihre Produkte keine Gewährleistung übernehmen. Ihre Software wird *as is* angeboten, mit so weitgehenden Garantie- und Haftungsausschlüssen, wie es die jeweilige nationale Jurisdiktion zuläßt.[560] Joseph Weizenbaum hat das Mißverhältnis zwischen der Software- und anderen Branchen einmal so beschrieben: eine Fluggesellschaft, die sich so verhalten würde wie eine Informatikfirma, wäre innerhalb von zwei Wochen pleite. Das Produkthaftungsrecht aus der Welt der materiellen Waren findet auf Software noch keine Anwendung, obgleich es wünschenswert wäre, eine Situation zu schaffen, in der Nutzer erwarten können, daß das, wofür sie bezahlt haben, auch funktioniert.[561] Das entspricht jedoch nicht dem Stand der Kunst in der Informatik, und auch der tatsächliche Trend geht in die entgegengesetzte Richtung. Das US-amerikanische Produkthaftungsrecht wird derzeit dahingehend korrigiert, daß den Herstellern von Software eine stark verminderte Haftung eingeräumt wird.[562] Es ist ferner zweifelhaft, ob die Gerichtsstandsklauseln wie sie in der Bankenwelt üblich sind, in denen Unternehmen eine Jurisdiktion der eigenen Wahl vorschreiben, in der Software-Branche Anwendung finden. Das Produkthaftungsrecht ist in verschiedenen Ländern sehr unterschiedlich geregelt, so daß Computer-Konzerne, auch wenn sie in den USA weitgehend aus der Haftung entlassen sind, in anderen Ländern mit sehr strikten Auflagen rechnen müssen.[563]

Umgekehrt verfügt die Welt der freien Software sehr wohl über Mechanismen der Qualitätskontrolle und der Stabilitätssicherung für offiziell freigegebene Releases. Der offene Prozeßcharakter und die fehlende Rechtsform vieler Projekte schließt jedoch eine

---

[560] Als Beispiel hier einige Passagen aus dem Microsoft *End-User License Agreement* (EULA). Darin wird zunächst betont, daß die Nutzerin durch Installation, Kopieren oder jegliche Form von Gebrauch der Software einen rechtsverbindlichen Vertrag mit Microsoft eingeht, und daß der Käufer keineswegs die Software erwirbt, sondern nur ein Nutzungsrecht lizenziert: “The SOFTWARE PRODUCT is licensed, not sold.” Mit der Lizenz willigt er in folgende Bedingungen über Garantie- und Haftungsausschluß ein:
“7. NO WARRANTIES. Microsoft expressly disclaims any warranty for the SOFTWARE PRODUCT. THE SOFTWARE PRODUCT AND ANY RELATED DOCUMENTATION IS PROVIDED "AS IS" WITHOUT WARRANTY OR CONDITION OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT REMAINS WITH YOU.
8. LIMITATION OF LIABILITY. In no event shall Microsoft or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use the SOFTWARE PRODUCT, even if Microsoft has been advised of the possibility of such damages. Because some states and jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.” (End-User License Agreement for Microsoft Software, DCOM98 for Windows 98, version 1.3, http://www.microsoft.com/com/dcom/dcom98/eula.asp)

[561] Wolfgang Coy, Fachgespräch 7/1999

[562] Dalheimer, Fachgespräch 7/1999

[563] Rössler, Fachgespräch 7/1999

rechtsverbindliche Gewährleistung aus. Das gleiche gilt auch für Firme, die freie Software zusammenstellen und vertreiben. SuSE z.B. bietet keine generelle Gewährleistung.[564]

Anders sieht es bei Firmen aus, die Dienstleistungen auf Basis freier Software anbieten. Da sie die Software, mit der sie arbeiten, kennen und im Sourcecode überprüfen können, können sie auch einschätzen, welche Leistungen sie gewährleisten können und welche nicht. LunetIX z.B. gewährleist sehr umfänglich, daß das, was mit dem Kunden vereinbart ist und wofür er bezahlt, auch tatsächlich zum Funktionieren gebracht werden kann. “Ich habe die Erfahrung gemacht, daß man viel ehrlicher ist, mit den Sachen die nicht gehen. Eine Freiheit, die wir uns nehmen können, weil wir eben kein Produkt verkaufen, das *all singing, all dancing* ist. Wir können sagen, da hat es seine Stärken, da seine Schwächen. Wenn die Schwächen für Euch ein Problem sind, können wir daran arbeiten.”[565]

Ingo Ruhmann ist der Ansicht, daß im Streitfall eine Sachverständigenüberprüfung vor Gericht nur dann überhaupt möglich ist, wenn der Quellcode eingesehen werden kann.

> “Das Problem an der Sache ist aber: Wieviele Fälle von Prozeßen gegen solche Firmen werden überhaupt geführt? Die Kriminalstatistik sagt, daß es jenseits von Kreditkartenbetrug allenfalls ein bis zwei Dutzend Fälle von Computerkriminalität gibt, die vor Gericht landen. Das rührt einfach daher, daß keine Analyse der Systeme möglich ist. Das heißt, wenn ein Fehler auftritt, kann ich mich effektiv nicht mit einiger Sicherheit an die Firma wenden, die für diesen Fehler verantwortlich ist. Wir kennen es, wenn man eine Hotline befragt, dann wird meistens gesagt: 'Das könnte ja auch noch den und den Grund haben. Es könnte der Treiber von irgendeinem Hardwareteil sein.' Solange die Sourcen nicht wirklich offenliegen, hat der Anbieter eines prorietären Systems natürlich die Möglichkeit abzustreiten, daß der Fehler bei ihm liegt. Und ich habe keine Möglichkeit, das herauszufinden.
>
> Bei Black-Box-Systemen ist eine Fehlererkennung nur durch einen Funktionsfehler im Betrieb möglich, bei Open Source-Systemen aber eine konkrete Analyse, woran dieser Fehler liegen könnte. Das heißt, effektiv ist erst in dem Fall, daß ich Open Source-Software einsetze, überhaupt die Möglichkeit gegeben, daß ich diesen Rechtsweg -- den viele bei prorietären Systemen für gegeben halten -- mit einiger Aussicht auf Erfolg beschreiten kann. Hinzu kommt sogar, daß ich, wenn ich bei bestimmten proprietären Systemen eine Analyse mache und herausfinde, daß der zur Sicherheit benutzte Mechanismus extrem schwach ist, da oftmals sogar rechtlich besonders schlechte Karten habe, weil der Schutz gegen Datenmißbrauch auch einen ausreichenden technischen Hintergrund voraussetzt. Das heißt, sind die technischen Mechanismen schwach, habe ich rechtlich auch gar keine Möglichkeiten. Das Ganze zu überprüfen ist rechtlich ausgesprochen schwierig, und möglich ist es nur bei Open Source-Systemen.”[566]

Eine systematische Kontrolle der Funktionstüchtigkeit von *Black Box*-Systemen ist ausgeschlossen. Eine gerichtliche Würdigung ist allenfalls möglich, wenn ein Unternehmen den Quellcode seiner proprietären Software (üblicherweise unter Vertraulichkeitsverpflichtung) den Sachverständigen zugänglich macht.

---

[564] Hohndel, Fachgespräch 7/1999

[565] Hetze, Fachgespräch 7/1999

[566] Ruhmann, Wizards 7/1999

Eine Überprüfung vorab ist selbst bei Vorliegen des Quellcodes schwierig, doch nachträglich lassen sich immer wieder bewußt eingebaute Sicherheitslücken nachweisen. Hier sieht Frank Rieger eine Möglichkeit für das Eingreifen des Gesetzgebers. Wenn die Haftbarkeit von Herstellern für Hintertüren drastisch verschärft würde, gäbe es keine anderen Möglichkeit mehr gibt, als Sicherheitsprodukte Open Source zu machen. "Dadurch würde das, was sowieso schon üblich ist bei kleineren Firmen, alles, was Krypto und Security ist, Open Source zu machen, auch für große Firmen bindend und verpflichtend werden, und könnte tatsächlich eine ganze Menge bewegen."[567]

Kryptografie: *Security by Obscurity* vs. offene Verfahren

Das Schlüsselelement für Sicherheit in digitalen Systemen ist die Kryptografie. Der Schutz von Daten auf einem Rechner (z.B. Patientendaten), die Vertraulichkeit von Kommunikationen (z.B. Vertragsverhandlungen zwischen Unternehmen), die Verhinderung der nachträglichen Veränderung von Dokumenten (z.B. vertraglichen Abmachungen), die Sicherung der Identität von Transaktionspartnern durch digitale Signaturen, der Beleg für das Absenden und den Empfang von Dokumenten, die Verhinderung der Nichtanerkennung von Vereinbarungen und schließlich der gesamte Komplex der digitalen Zahlungsverfahren beruhen alle auf kryptografischen Systemen. Sie waren bis vor kurzem die exklusive Domäne von Militär und Geheimdiensten. Als in den 70er Jahren Konzerne und Banken begannen, digitale Netze einzusetzen, entstand erstmals ein ziviler Bedarf nach Kryptografie. Seit Anfang der 90er Jahre verbreitet sich die PC-gestützte private Nutzung von offenen Netzen und damit der allgemeine Bedarf nach kryptografischen Produkten.

Zwei Positionen treffen an dieser Frage aufeinander: einerseits das Grundrecht auf Unantastbarkeit der Kommunikation und Schutz der Privatsphäre, andererseits das Sicherheitsinteressen des Staates. Die Konfliktlinie verläuft in allen Ländern zwischen den Daten- und Persönlichkeitsschutzverfechtern und den Strafverfolgungs- und Nachrichtendiensten, zwischen den Wirtschafts- und den Innenministerien.[568] Schließlich ist Kryptografie nicht nur Voraussetzung für eCommerce, sondern selbst ein erheblicher Markt.

Wenn Kryptografie Vertrauen in das Internet schaffen soll, stellt sich die Frage, auf welcher Grundlage man Kryptografie-Software vertrauen kann. Auf dem Markt für diese Produkte stehen sich zwei Herangehensweisen gegenüber. Zum einen soll ein Vertrauen in die Sicherheit solcher Systeme erzeugt werden, indem sie in einem vertraulichen Prozeß entwickelt werden und ihre Funktionsweise geheimgehalten wird. Dieser sog. *Security by Obscurity*-Ansatz herrscht unter den proprietären oder *Black Box*-Produkten vor. Dagegen kann die Sicherheit von Open Source-Systemen, die im Prinzip jeder Interessiert einsehen kann, nur darauf beruhen, daß der Mechanismen auch dann schützt, wenn er bekannt ist.

> "Es gibt einige Microsoft-Systeme, die ihre Paßwörter ganz einfach mit einer xor-Verschlüsselung mit einer relativ einfachen Zahl ablegen. Wenn bekannt wird,

---

[567] Rieger, Wizards 7/1999-Diskussion

[568] Ein Versuch, diesen Konflikt im Interesse der Sicherheitsbehörden zu lösen, war die Clipper-Chip-Initiative von 1993 (http://www.eff.org/pub/Privacy/wh_crypto_original.announce). Sie wurde von den Verfechtern des Persönlichkeitsschutzes und vom Markt zurückgewiesen (s. Froomkin http://www.law.miami.edu/~froomkin/articles/planet_clipper.htm und EPIC http://www.epic.org/crypto/clipper/)

> wie dieser Mechanismus funktioniert, dann ist er nichts mehr wert. Im Gegensatz dazu, wie allen bekannt, verschlüsselt Unix Paßwörter in einer Einwegfunktion und vergleicht sie auch in der verschlüsselten Version. Da es keine Umkehrfunktion dazu gibt, ist es völlig unerheblich, ob diese Einwegfunktion bekannt ist oder nicht. Und jeder kann auch das abgelegte Paßwort lesen, weil er damit nichts anfangen kann, denn es liegt ja nicht im Klartext vor."[569]

Rieger nennt als weitere Beispiele GSM, den Standard, der heute in den meisten Mobiltelefonen verwendet wird, und der trotz strenger Geheimhaltung von einer US-amerikanischen Hacker-Gruppe geknackt werden konnte, sowie die Telefonkarte der Deutschen Telekom. Im letzteren Fall kam ein Hamburger Sicherheits-Consulting-Unternehmen zu dem Ergebnis, daß die von der Telekom gewählte Chipkarte keine inhärente Sicherheit biete, sondern diese allenfalls in einem Technologievorsprung gegenüber möglichem Mißbrauch von etwa drei oder vier Jahre beruhe. Die Vorhersage hat ich bewahrheitet. Gefälschte Telefonkarten sind ein erhebliches Geschäft. Rieger schätzt, daß es sich um einen der größten Schäden handelt, die durch Security by Obscurity entstanden sind. Er bezeichnet das Modell der *Security by Obscurity* als ein Wissensmonopol einer Priesterschaft:

> "Ich designe ein System so, daß es von außen auf den ersten Blick nicht durchschaubar ist und seine inneren Wirkungsweisen nicht verstanden werden können -- so glaubt man zumindest. Und wenn man dann der Sache doch nicht so traut, baut man lieber noch so ein paar Gemeinheiten ein, einige Fallen oder Inkonsistenzen, um zu verhindern, daß, falls es einen Security-Bruch gibt, die Sachen zu schnell vorangehen. Dieses geschlossene Modell ist sehr kompatibel mit dem Kulturraum von großen Entitäten wie Siemens z.B. ... Die haben da eine kleine Abteilung mit einer großen Stahltür davor, und da ist ein Zahlenschloß und ein Iris-Scanner dran und noch eine große Stahltür und dahinter liegen in dem Safe die drei Blätter Papier, auf denen steht, wie's geht. Da ist etwas, was sie verstehen. Da können sie einen Wachmann davorstellen, der notfalls schießt, und damit ist die Sicherheit dann eigentlich gewährleistet. Daß der Algorithmus, den sie in dem Panzerschrank mit den drei Stahltüren haben, auf jeder beliebigen Chipkarte, die zehn Millionen Mal verteilt wird, nach draußen geht, daran haben sie nicht wirklich gedacht."[570]

Das Priesterschaftsmodell wird ferner durch die zunehmende Mobilität und Informationsdichte infrage gestellt. Angestellte aus Sicherheitsabteilungen wechseln ihre Arbeitgeber häufiger als früher, und kein *Non-Disclosure Agreement* kann verhindern, daß sie ihr Wissen mitnehmen.

Das quelloffene Modell ist wiederum nicht per se sicherer, doch durch das öffentliche Testen und Studieren werden Mängel und Fehler meist relativ schnell gefunden und behoben. Auch Hintertüren sind dadurch nicht ausgeschlossen, doch auch hier ist die Chance, daß jemand sie entdeckt, relativ hoch.

Rieger weist aber darauf hin, daß Open Source kein Allheilmittel ist. Besonders große Programme, die erst nachträglich quelloffen gemacht werden, sind extreme schwer zu

---

[569] Ruhmann, Wizards 7/1999

[570] Rieger, Wizards 7/1999

überschauen. Kaum jemand hat den gesamten Quellcode von PGP durchgelesen und kann sicher sagen, daß sich darin keine Hintertür befindet. Und wenn man so etwas mit mehreren Personen macht, ist die Gefahr, etwas zu übersehen, noch viel größer. Seit Anfang 1998 liegt das Programm PGP 5.0 im Quelltext vor, ein gravierender Fehler, der unter bestimmten Voraussetzungen die Sicherheit unterminiert, für die das Programm eigentlich sorgen soll, wurde jedoch erst im Mai des darauffolgenden Jahres bekannt.[571] Für Rieger ist der einzige Weg zu solchen Kernsicherheitssystemen wie PGP, diese von Anfang an offen zu entwickeln. Nur ein permanent mitlaufender *Review* bietet Gewähr, daß nicht nachlässig oder gezielt Sicherheitsprobleme eingebaut werden.[572]

Die Bundesregierung faßte am 2. Juni 1999 einen Beschluß, der die Verwendung von starken Kryptografieprodukten ohne jede Beschränkung vorsieht.[573] Sie strebt damit ihren breiten Einsatz durch jedermann an. Zu den Schritten, die Hubertus Soquat (IT-Sicherheit im BMWi) in Berlin vortrug, gehört die Erhöhung der Verfügbarkeit von Kryptoprodukten, von der Entwicklung über die Produktion und den Vertrieb bis zum Einsatz. Im Zentrum der Aktivitäten zur Sensibilisierung breiter Nutzerschichten steht die Kampagne "Sicherheit im Internet",[574] die sich besonders an normale Nutzer und an kleine und mittlere Unternehmen richtet. Das BMWi wird die deutsche Krypto-Industrie beobachten, um abzuwägen, welche Möglichkeit bestehen, durch öffentliche Auftraggeber entsprechende Nachfragepotentiale zu realisieren. Zur Förderung des Einsatzes von Verschlüsselung innerhalb der Bundesbehörden und darüber hinaus dient das Pilotprojekt "Sphinx".[575]

Auf die Frage, was geschehe, wenn sich herausstellt, daß das, was die Strafverfolgungs- und Nachrichtendienste als ihre legitimen Bedürfnisse sehen, sich mit alle anderen Eckpunkten ausschließt, wie z.B. *Security*, sicherer Nachrichtenverkehr, freie Verfügbarkeit der starken Kryptographie, antwortete Soquat in einem TAZ-Interview:[576] “Dann haben sie [die Sicherheitsdienste] Pech gehabt.” Er verwies darauf, daß Nachrichtendienste und die Strafverfolgungsbehörden nachgeordnete Behörden sind und daß das Bundesverfassungsgerichtes in einer jüngsten Entscheidung unterstrichen hat, daß es Grenzen für deren Tätigwerden gibt. Wenn die Maßnahmen des BMWi nach Ablauf von zwei Jahren 2001 einer Prüfung unterzogen werden, spielen dabei neben internationalen Fragen auch die Interessen der Strafverfolgungs- und Sicherheitsbehörden eine Rolle.

Voraussetzung für das Vertrauen in Kryptografie ist also einerseits, so auch Soquat, die Quelloffenheit der Software. Andererseits strebt die Bundesregierung eine Prüfungen und Zertifizierungen von Software an, die u.a. vom Bundesamt für Sicherheit (BSI) auf Grundlage eines Erlasses des Bundesinnenministeriums vorbereitet wird.

Erschwert wird die Situation dadurch, daß Kryptgrafie in vielen Ländern als Waffentechnologie klassifiziert und mit Ein- und/oder Ausfuhrbeschränkungen belegt ist.[577]

---

[571] http://www.securiteam.com/securitynews/Key_Generation_Security_Flaw_in_PGP_5_0.html

[572] Rieger, Wizards 7/1999

[573] Ernst Sandl, zuständig für IT-Sicherheit im im BMWi, hat diese Vorlage maßgeblich mit vorbereitet.

[574] http://www.sicherheit-im-Internet.de/

[575] http://www.bsi.bund.de/aufgaben/projekte/sphinx/index.htm

[576] und wiederholte es in der Wizards 7/1999-Diskussion

[577] zur jeweiligen Politik in den einzelnen Ländern s. Koops http://cwis.kub.nl/~frw/people/koops/lawsurvy.htm. Die USA, die nicht nur die Ausfuhr starker Kryptografietechnologie verboten, sondern auch eine aggressive Außenpolitik betrieben, um alle anderen

Das führt dazu, daß Hersteller von kryptografischen Algorithmen oder anderer Software, die solche Algorithmen enthält, jeweils eine starke Version für den Inlandsmarkt und eine schwächere für den Export anbieten müssen. Freie Software-Projekte, die sich nicht nach Ländergrenzen organisieren, stehen hier vor besonderen Problemen.

Das Wassenaar Abkommen[578] ist ein Versuch, die verschiedenen nationalen Politiken zu harmonisieren. Für die freie Software besonders interessant ist, daß es kryptografische Software in der *public domain* von Restriktionen ausnimmt.

> "Nun profitiert als 'public domain' -- in einem speziellen Sinne, der eigentlich heißt: frei weitergebbar -- gekennzeichnete Software von Teil 2 der *General Software Note* des Wassenaar Arrangements. Es gibt relativ glaubwürdige Hinweise, u.a. in Form der neuen australischen Exportrichtlinien, daß wohl darauf gedrungen wird, diese Ausnahme für kryptographische Software aus dem Wassenaar Arrangement zu streichen. Das hieße, daß man diese kryptographische Software innerhalb der EU im kommenden Jahr [2000] zwar noch frei verteilen könnte, aber international nicht mehr. Wenn ich das mit Aspekten, wie der von Herrn Dalheimer angesprochenen großen Internationalität schon in der Software-Entwicklung, dem generisch zu dieser Software gehörenden weltweiten Zurverfügungstellen des Quellcodes, in einen Topf rühre, dann bekomme ich da heraus: einen Angriff auf die legale Erstellbarkeit frei verfügbarer Sicherheits-Software. Angenommen, diese freie Software wäre nicht mehr per se ausgenommen von den Restriktionen, dann kann ich zwar versuchen, diese Exportrestriktionen zu umgehen oder zu ignorieren (was mit Sicherheit auf viele machen werden), aber bei dem, wovon wir hier reden: Einsatz freier Software in Verwaltungen und Unternehmen, da ist das nicht mehr eine Option. Da muß ich schon davon ausgehen können, daß diese Sachen in einem legalen Rahmen erstellt wurden und erhältlich sind."[579]

Die Kryptorichtlinien der Bundesregierung haben international Anerkennung gefunden, sind aber auch, z.B. in den USA auf Kritik gestoßen. Auch Frankreich und Großbritannien haben eine andere Haltung dazu als die die Bundesregierung. Diese will aber in den laufenden Gesprächen im Rahmen des Wassenaar-Abkommens selbst, seiner Umsetzung innerhalb der EU auf *Dual-Use*-Verordnungsebene und bei der Umsetzung dieser europäischen Regelungen auf der nationalen Ebene dafür einsetzen, daß die Grundidee hinter den "Kryptografie-Eckwerten", also die freie Verfügbarkeit, nicht verwässert wird.[580] In einem ersten Schritt ist der Handel zwischen den Mitgliedsstaaten bereits liberalisiert worden.

Auch die NSA hat ihre Haltung gegenüber Verschlüsselungssystemen inzwischen abgeschwächt. Grund könnte sein, daß deutlich geworden ist, daß nicht alle Kryptografie-Kompetenz in den USA sitzt und selbst dafür Exportverbote wenig Wirkung zeigen. Die neue Strategie der NSA, so vermutet Rieger, sind Hintertüren.

Länder davon zu überzeugen, ebenfalls eine Schlüsselhinterlegungstechnologie (*Key eschrow*) einzuführen, haben vor kurzem ihre Exportvorschriften geändert, so daß starke Kryptografie ausgeführt werden darf, solange der Quellcode offen ist.

[578] http://www.wassenaar.org/

[579] Rössler, Fachgespräch 7/1999

[580] Soquat, Fachgespräch 7/1999

> “Darin haben sie eine wirklich große Erfahrung, angefangen bei der Krypto-AG, wo über Jahrzehnte hinweg die Keys mit im Cyphertext-Stream ge-leakt wurden, bis hin zu Hintertüren, die in irgendwelche Produkte eingebaut werden, z.B. Router, Firewalls, ISDN-Systeme, Betriebssysteme. Da wird, denke ich, in nächster Zukunft deren großes Arbeitsfeld liegen, weil sie genau wissen, daß der Aufwand, die Kryptografie auf dem Transportweg zu brechen, sich so schnell nicht verringern wird, selbst wenn sie noch so viel in die Forschung stecken. Deswegen ist der Weg, den sie gehen werden, Hintertüren. Wir haben auch einen deutlichen Anstieg von Angriffen über Hintertüren gesehen, die von professionellen Industriespionen erfolgen, die solche Hintertüren tatsächlich auch kaufen. Es gibt Leute, die kennen die halt und verkaufen dieses Wissen. Etliches von dem, was an Sicherheitsproblemen da ist, wird mittlerweile nicht publiziert, sondern immer noch unter der Decke gehalten, gerade was ISDN-Systeme betrifft.”[581]

Sicherheit läßt sich nicht auf die Frage der Kryptographie reduzieren.

> “Die eigentlichen Probleme in der Sicherheit liegen darin, daß man Software und Systeme mit sehr vielen Unbekannten verwendet. Bei der Kryptographie ist es so, ich kann mir vorstellen, ich habe eine starke Kryptographie unter MS NT über einen Treiber eingebunden und die Gefahren liegen wo ganz anders: die Dateien werden zwar wunderbar verschlüsselt, aber über einen verdeckten Kanal kommen sie letztendlich doch dort hin, wo sie nicht hin sollen. Aus diesem Grund ist der Schwerpunkt bei unserem Projekt nicht in der Kryptographie. Wir binden die hardware-mäßig ein. Das ist im Grunde nur eine Schnittstelle zu einer Hardware. Die müssen wir natürlich nicht offenlegen -- die Schnittstelle schon, aber die Hardware nicht. Selbst wenn wir ein Software-Modul schaffen würden unter Linux, auch das würden wir nicht weitergeben. Das würde mit eingelinkt werden in das System. Das ist so etwas wie ein Gerätetreiber oder etwas ähnliches. Und dort würde sich die Krypto befinden. Das ist überhaupt nicht der Sicherheitsfaktor. Sondern: die Tatsache, daß ich weiß, daß meine Krypto überhaupt angesprochen wird und daß nicht noch etwas anderes auf dem System gemacht wird -- all dieses sind die entscheidenden Sicherheitsfragen. Insofern laufen das Wassenaar-Aggreement und diese Dinge für mich im Grunde an der Zeit vorbei. Das ist nicht mehr das Problem. Und das weiß auch letzendlich die amerikanische Regierung, deshalb setzen sie auch viel mehr auf so etwas wie MS, da haben sie viel mehr Power drin, wenn die NSA dort ihren Einfluß geltend macht, als wenn sie den verlorenen Krieg gegen irgendwelche kryptographischen Programme führen -- das ist mit dem Internet sowieso vorbei. Insofern ist das nicht das Hauptproblem der Security. Und ich denke, wir haben das erkannt.[582]

Für die Verschlüsselung von Mail mit public-private-key-Verfahren gibt es derzeit zwei Standards: PGP (Pretty Good Privacy von Phil Zimmerman) und S-MIME. S-MIME setzt für die X.509-Zertifikate eine zentrale *Certificate Authority* voraus, die alle Schlüssel beglaubigt.

---

[581] Rieger, Wizards 7/1999

[582] Rudeloff, BSI, Fachgespräch 7/1999

Auch Open-PGP ist mittlerweile durch das RFC 2440 standardisiert. Es ist unter Zusammenarbeit von Lutz Donnerhacke aus Jena und Jon Callas von PGP erabeitet und verabschiedet worden, so daß auch hier der gleiche Status erreicht ist wie bei S-MIME. Die Zertifizierung von PGP-Schlüsseln beruht auf einem *Web of Trust* ohne zentrale oberste Instanz. Es erlaubt ebenfalls, hierarchische Strukturen für die Zertifizierung zu bilden, doch die Grundstruktur ist ein Netzwerke von Leuten, die sich kennen und vertrauen.

PGP weist zwei Probleme auf. Erstens steht es nicht unter der GPL. Das heißt, der Source Code ist zwar verfügbar, aber es ist nicht erlaubt, Modifikationen daran vorzunehmen und zu vertreiben, also zusätzliche Features einzubauen, Teile aus PGP zu entfernen, die man für Sicherheitsrisiken hält und Software bereitzustellen, die auf PGP basiert. Zweitens wurde PGP von Network Associates gekauft. Als kommerzielles amerikanisches Produkt unterliegt es den Exportbeschränkungen. Starkes PGP ab einer bestimmten Schlüssellänge darf nicht ausgeführt werden. Außerdem war Network Associates Mitglied der *Key Recovery Alliance*, einem Zusammenschluß von US-Firmen, der es ermöglichen soll, eine *Key Recovery*-Infrastruktur nach den Vorstellungen der US-Regierung zu betreiben, die nur dann an eine Firma Aufträge vergibt, wenn sie Mitglied in der *Key Recovery Alliance* ist. Dann sind Network Associates wieder ausgetreten und wieder eingetreten. Phil Zimmerman behauptet zwar, es gäbe eine klare Position gegen *Key Recovery*, aber die Firma, der diese Software gehört, hat offensichtlich eine andere Meinung.

In der jüngsten Entwicklung von PGP sind Features hinzugekommen, die sicherheitstechnisch bedenklich sind und die Rückwärtskompatibilität einschränken. Einer der Kritikpunkte betrifft das *Corporate Message Recovery* (CMR). Anders als beim firmenexternen *Key Recovery*, das die NSA anstrebt, wird dabei jede Mail an einen Angestellten einer Firma nicht nur mit dessen persönlichem öffentlichem Schlüssel, sondern auch mit dem seines Unternehmens verschlüsselt. Ziel ist, daß die Firma auch dann noch auf die Korrespondenz dieses Mitarbeiters zugreifen kann, wenn er selbst nicht mehr zur Verfügung steht. Andreas Bogk (CCC) sieht darin die Unterscheidung zwischen der Lagerung und der Übertragung von Informationen verletzt.

> "Eigentlich viel sinnvoller wäre es, einen Mechanismus einzubauen, der einen sogenannten *Storage Key* verwendet. Das heißt, wenn etwas an mich geschickt wird, lege ich das mit einem *Storage Key* ab, auf den die ganze Abteilung Zugriff hat. Es sind auch Verfahren denkbar, bei denen Abteilungsschlüssel existieren, mit denen jeweils zwei Personen aus der Abteilung zusammen die Mail eines dritten lesen können usw. Also, die Art, wie diese *Corporate Message Recovery* angewendet werden soll, ist nicht wirklich nachvollziehbar. Aber sie ist natürlich hervorragend dafür verwendbar, zu einem späteren Zeitpunkt eine Überwachung einzurichten, indem man einfach bestimmten Firmen vorschreibt, eine solche *Corporate Message Recovery* mit einem Schlüssel zu verwenden, der bei staatlichen Stellen hinterlegt ist."[583]

Eine weitere bedenkliche Neuerung ist die *Message-ID*, die seit PGP-Version 5.0 unverschlüsselt im Header steht. Offiziell soll sie dazu dienen, eine Mail, die in mehreren Teilen verschlüsselt ist, wieder zusammenzusetzten, doch existiert für diese Aufgabe der technische Standard Multipart-MIME. Vollkommen unbegründet ist diese *Message-ID* in

[583] Andreas Bogk, Wizards 7/1999

einer PGP-Message, die nur aus einem Teil besteht. Bogk sieht darin einen weiteren Weg, über den eine PGP-Nachricht angegriffen werden kann.

> “PGP ist ja ein Hybridverfahren, d.h., ich habe einmal ein *Public Key*-Verfahren, mit dem ein *Session Key* verschlüsselt wird. Mit diesem *Session Key* wird dann in einem klassischen symmetrischen Verfahren der Rest der Mail verschlüsselt. Die beiden Punkte, die ich angreifen kann, sind einmal das *Public Key*-Verfahren. Wenn ich es schaffe, aus dem *Public Key* den *Secrete Key* zu generieren, kann ich die Message lesen. Der zweite Punkt, den ich angreifen kann, ist der *Session Key*, der für den symmetrischen Schlüssel verwendet wird. Wenn ich den knacke, kann ich die Message auch lesen. Und diese *Message-ID* bringt nun einen dritten Angriffsweg mit. Der *Session Key* wird nämlich aus einem Entropie-Pool generiert. Da werden also Random-Informationen gesammelt, miteinander ver-*hasht*, und daraus wird der *Session Key* gewonnen. Und direkt nachdem der *Session Key* generiert wurde, wird diese *Message-ID* aus demselben Pool generiert. Da ist zwar ein Prozeß dazwischen, der sich *Whitening* nennt und der mit Hilfe eines *Hashing*-Algorithmus' verhindern soll, daß man daraus den Zustand des Pools zurückrechnet, aber erstens ist die Forschung auf dem Gebiet noch ein bißchen dünn und zweitens wird hier grundlos ein weiterer Angriffsweg eingeführt.”[584]

Inkompatibilitäten in der PGP-Welt schließlich hat die Umstellung des zugundeliegenden *Public-Private-Key*-Algorithmus’ bewirkt. Das bislang verwendete RSA-Verfahren hat den Nachteil, patentiert zu sein. Das Patent auf dem jetzt in PGP verwendeten Diffie-Hellman-Verfahren ist 1998 ausgelaufen. Dieser Schritt, der die Zugänglichkeit der Technologie erhöht, führt dazu, daß diejenigen, die PGP 6.0 verwenden, nicht mehr mit denjenigen kommunizieren können, die ältere Versionen benutzen.

Aus diesen Gründen wurde das freie Projekt GNU-Privacy-Guard[585] ins Leben gerufen. Eine Gruppe um Werner Koch hat dazu den Open PGP-Standard neu implementiert. GNU-PG kann sowohl mit alten als auch mit neuen PGP-Versionen kommunizieren. Es enthält kein *Corporate Message Recovery* oder *Message-IDs.* Und schließlich steht es unter der GPL, sodaß sichergestellt ist, daß der Quellcode überprüfbar bleibt und jeder die Teile, die ihm verdächtig vorkommen, entfernen kann. Als "public domain"-Software ist es von den Restriktionen des Wassenaar-Abkommens ausgenommen. “Durch diesen Open-Source-Ansatz ist eine dezentrale Evaluierung durch zahlreiche Stellen und Nutzer auch für gehobene Ansprüche realisierbar. ... Mit dem Konzept von GPG könnte ein Werkzeug geschaffen werden, das ohne Einschränkungen für alle Benutzerschichten frei und unentgeltlich verfügbar ist (inkl. Behörden, kommerzielle Nutzer, Privatbenutzer etc.).”[586]

Im Juli 1999 lag GNU-PG in einer *Command Line*-Version und als Integration in Mutt (einer der beliebtesten Unix-Mail-Clients) und Emacs vor. Was noch nicht existierte, waren Einbindungen in klassische Mail-Programme wie Eudora und Netscape. Ziel des Projektes ist es, mehr Internet-Benutzern, vor allem auch denjenigen in der Windows-Welt, GNU-PG zugänglich zu machen.[587]

---

[584] Bogk, Wizards 7/1999

[585] http://www.gnupg.org/

[586] Soquat, Open Source und IT-Sicherheit, Text zu Wizards 7/1999

[587] Bogk, Wizards 7/1999

Es ist das erklärte Interesse auch des BMWi, daß GNU-PG möglichst breit eingesetzt wird und dazu komfortablerer Benutzerschnittstellen und eine Einbindung in die verschiedenen Betriebssysteme und eMail-Programme geschaffen wird. Auch hier gibt es inzwischen erste Ergebnisse. Das BMWi fördert daher die Weiterentwicklung des GPG mit 318.000 DM.[588]

Militärische Software und Hochsicherheitsbereiche

Die Sicherheitsanforderungen sind beim privaten und wirtschaftlichen Gebrauch andere als in sensitiven Bereichen, wie Banken, Atomkraftwerken, Militär und Nachrichtendiensten. Selbst in diesen Einsatzfeldern gewähren Software-Unternehmen ihren deutschen Kunden keinen Einblick in den Quellcode. Für den Einsatz im Hochsicherheitsbereich evaluiert das BSI Open Source-Software. Andere, wie der Hamburger Informatiker Klaus Brunnstein, hingegen halten OSS für nicht hochsicherheitstauglich.

In militärischen Einsatzbereichen geht seit einigen Jahren sowohl in den USA wie in Europa der Trend weg von spezialisierter Software. 1994 gab US-Präsident Clinton im Zuge einer Kostenreduzierung die Anweisung aus, daß Behörden wo immer möglich *Commercial Off the Shelf* (COTS)-Software einsetzen sollten. Für spezialisierte Anwendungen in Bereichen wie Militär oder Banken werden *Customized Automated Information Systems* (CAIS) eingesetzt. Auch das im Januar 1997 gestartete Programm *Information Technology for the 21st Century* (IT-21) sieht den Übergang zu einem PC-gestützten taktischen und unterstützenden Kriegsführungsnetzwerk und dabei zu Industriestandards vor. Ziel von IT-21 ist es, alle US-Streitkräfte und letztendlich auch die amerikanischen Allierten mit einem Netzwerk zu verbinden, das ein nahtlose Sprach-, Video- und Datenübertragung von geheimer und nichtgeheimer, taktischer und nichttaktischer Information über dieselben PCs erlaubt. Konkret heißt dies, daß die gemeinsame Betriebsumgebung der US-Streitkräfte auf Windows NT, MS Exchange und MS Office beruhen soll. Die Verwendung aller Nicht-Standard-Netzwerkbetriebssysteme und eMail-Produkte (wie das DoD *Messaging System* AUTODIN) wurde zum Dezember 1999 eingestellt.[589] Software-Entwicklung für den ausschließlich militärischen Einsatz (*Military Unique Development*) macht einen zunehmend kleineren Anteil aus, der aber als wichtigste Kategorie für die nationale Sicherheit angesehen wird (Waffensysteme, störsichere Kommunikation und Atombombensimulationen). Selbst für diese Katergorie von Software werden Stimmen lauter, die den Basar als das beste Entwicklungsmodell ansehen.

In seinem Aufsatz “Open Source and these United States” zählt C. Justin Seiferth, Major der US Air Force, die Vorteile auf, die das US-Verteidigungsministerium (DoD) daraus ziehen kann, wenn es generell eine Open Source-Lizenzierungspolitik betreiben würde. Dadurch könnten Kosten gesenkt und die Qualität der Systeme und die Geschwindigkeit, mit der sie entwickelt werden, erhöht werden. Die Zusammenarbeit mit anderen Organisationen, Händlern und Alliierten werde durch quelloffene Software erleichtert. Das gleiche gelte für die Beziehungen zwischen Behörden unter dem DoD, da sie nicht auf finanziellen Transaktionen beruhen, sondern auf Tausch und Management-Abkommen. Da viele Entwickler die Gelegenheit, interessante Arbeit zu leisten, höher

---

588 http://www.sicherheit-im-internet.de/showdoc.php3?doc=bmwi_min_doc_1999944125721&page=1

589 Clemins o.J.

schätzten als finanzielle Entlohnung, sei quelloffene Software auch ein Anreiz, IT-Spezialisten im DoD zu halten. “The adoption of open source licensing may allow the military to leverage some of the current enthusiasm garnered by open license related methods. ... The judicious application of open licensing offers the possibilities of improving both the performance of government systems and the job satisfaction, competence and retainability of military members, civilians and contractors.”[590] Die Hauptvorteile einer Übernahme der Open-Source-Konzepte für das DoD sieht Seiferth in der Interoperabilität, im langfristigen Zugang zu archivierter Information und in einem Kostenfaktor, der bislang verhindert habe, daß das Militär auf dem neuesten Stand der Software-Entwicklung bleibt.[591] Offene Dokumentation könne für den jeweiligen Anwendungsbereich angepaßt werden. Zur Frage der Sicherheit schreibt er:

> “While many non-technical managers believe the release of source code lowers the security of a system, experience shows the opposite. Security ‘holes’ are omissions or weaknesses designed into the code. Compiling the code into a binary application doesn't fix a security hole or hide it from prying eyes. Unfortunately, the discovery of all security problems is known in computer sciences as an NP-Hard problem- that is one which is believed to be impossible to resolve absolutely. The best current practice can hope for is to avoid obvious mistakes, test as much as a project's budget can afford and correct problems as soon as they are identified. Open licensing is designed to tackle this process. Most security experts believe that the release of source code improves, rather than diminishes the security of a software system. As the saying goes, bad news doesn't improve with time. It is far better to go ‘open kimono’ and identify security risks early when there is a better chance they can be fixed with less schedule and cost risk. Further, if components supplied by foreign or unfamiliar subcontractors are incorporated into the system, open licensing makes it far less likely that an accidental or deliberate security problem will be introduced. ... Open licensing also ensures that an identified fix can be incorporated without being hindered by licensing arrangements or proprietary agreements. If one of configuration or operation and procedures are required to prevent a vulnerability, open licensing allows the manuals and technical orders to document the vulnerability.”[592]

Es gäbe bereits Open Source-Aktivitäten, auf die eine breitere Übernahme innerhalb des DoD aufsetzen könnte. “Within the Department of Defense, the National Laboratories and Defense Advanced Research Agency have been the most visible users and producers of open licensed systems. They've released such advances as the original firewall and network security toolkits. As a more recent example, within the last year the National Laboratories have been constructing inexpensive supercomputers.”[593] Er bezieht sich dabei auf die FORTEZZA-Algorithmen der NSA, die im *Defense Messaging System* verwendet werden, und auf den Firewall-Toolkit der DARPA, als zwei extrem erfolgreiche Beispiele dafür, wie eine offene Lizenz die Akzeptanz und die Qualität von Systemen im Sicherheitsbereich dramatisch

---

[590] Seiferth 1999

[591] so waren z.B. Telefonvermittlungsanlagen nicht Y2K-fest, doch erst der herannahende Jahrtausendwechsel ermöglichte die Investition von $70 Millionen für die Erneuerung der Software.

[592] Seiferth 1999: 43

[593] Seiferth 1999: 8

erhöhen kann.[594] Ferner führt er die Forschungsförderung für eine der größten Erfolgsgeschichten des Open-Source, das Internet-Protokoll TCP/IP, an. Mit einem jährlichen IT-Haushalt von mehr als $57 Milliarden stelle die US-Regierung ein bedeutendes Marktgewicht dar, das sie verwenden könne, um auch kommerzielle Anbieter zu ermuntern, ihre Produkte unter offene Linzenzen zu stellen.

Vertrauenswürdige Instanzen

Software-Systeme haben mittlerweile eine Komplexität erreicht, daß nur wenige Experten sie noch durchschauen können, und gleichzeitig stellen sie einen allgemeinen Massenmarkt dar. Es besteht also eine Dichotomie zwischen einer Minderheit von Experten und der Mehrheit von Benutzern, die zwar im Prinzip die Möglichkeit hat, Open Source-Systeme zu überprüfen, effektiv hat sie sie aber nicht. Für eine Mehrheit ist -- bei quelloffener ebenso wie bei proprietärer Software -- das Funktionsprinzip von IT-Sicherheit nichts als der Glaube, daß es funktioniert. Der Mehrheit ist es nur möglich, auf das Urteil von Experten zu vertrauen.

> "Wenn ich irgendein System beschaffe, und ich bin mir nicht klar, wie sicher das ist, kann ich mir eine Sicherheitsfirma anheuern, die mir das erklärt und die ich selber dafür bezahle. Das ist die eine Lösung. Die andere Möglichkeit wäre, daß Distributoren dafür zahlen, daß sie sichere Systeme haben und das jemanden haben prüfen lassen. Das ist die kommerzielle Variante. Oder, und das wäre vielleicht das Effektivste, diese vertrauenswürdigen Instanzen beteiligen sich direkt an der Entwicklung.
>
> Wer könnten diese Instanzen denn sein? Es gibt, und das ist das interessante an offenen Systemen, auf einmal doch eine ganze Reihe mehr Akteure, als bisher. Bisher hatten wir für solche Überprüfungen ein Standardmodell: Der TÜV-IT oder das BSI bekommt einen Antrag auf den Tisch, ein bestimmtes System auf einen bestimmten Sicherheitsgrad hin zu prüfen. Die bekommen auch den Quelltext, machen hinterher einen Stempel drauf, nachdem sie Gebühren kassiert haben, und das war's. Wenn ich Open Source-Software habe, kann ich mir diese Sicherheitsakteure für entsprechende Zwecke, für die ich die Software hinterher einsetzen will, aussuchen. Das können entweder staatliche Akteure sein, d.h. Datenschutzbeauftrage, wenn es um Datenschutzfragen geht, das BSI oder TÜV-IT wie herkömmlich, oder ich kann mir auch den CCC vorstellen oder irgendwelche anderen Instanzen, die hier gerufen werden, um die Sicherheit bestimmter *Features* und Systeme zu überprüfen.
>
> Man kann sich genauso gut vorstellen, daß diese vertrauenswürdigen Instanzen national gerufen werden oder, wenn ich ein System global vertreiben will, daß ich mir international verteilt verschiedene Instanzen ins Boot hole. Und man könnte sich sogar vorstellen, daß hier neue organisatorische Strukturen geschaffen werden, im Zusammenhang mit der Open Source-Bewegung z.B. eine *Security Task Force* -- das ist nur so ein hingeworfener Begriff, der in diesem Bereich recht häufig zu hören ist -- zur Evaluation von Sicherheits-*Features*. Genau so etwas wäre eine Möglichkeit, um eine Vielzahl von Sicherheitsexperten in die Entwicklung miteinzubeziehen. Ich denke, der Beitrag von Frank Rieger hat auch gezeigt, daß eine

[594] Seiferth 1999: 34

> ex-post-Analyse eines Riesenberges von Source Code genauso schwierig ist für solche Experten, wie für das BSI, wie für jeden anderen. Wichtiger und besser ist, diese Experten gleich in den Entwicklungsprozeß miteinzubeziehen."[595]

Bei den vertrauenswürdigend Instanzen ist eine gewisse Vielfalt gefordert, da eine Instanz, die für eine Person oder eine Firma vertrauenswürdig ist, es nicht unbedingt auch für andere ist. Ein Beispiel aus einem anderen Bereich ist die Stiftung Warentest, die anhand von Kriterien prüft, die allen offengelegt werden. Die Enquetekommission "Deutschlands Weg in die Informationsgesellschaft" des deutschen Bundestages in der letzten Legislaturperiode hat in ihrem Bericht zu IT-Sicherheit eine "Stiftung Software-Test" gefordert, die sich mit den Kriterien und mit der Evaluation von Software unter IT-Sicherheitsaspekten auseinandersetzen sollte.

Statt einer zentralistischen "Stiftung Software-Test" wäre auch ein Netzwerk von vertrauenswürdigen Instanzen vorstellbar. In jedem Fall ist deutlich, daß eine begründete Vertrauensbildung eine komplexe Sicherheitsinfrastruktur erfordert. Dokumentation des Quellcodes und Werkzeuge, die sein Studium erleichtern, würden eine Überprüfung auch von zertifizierter Software ermöglichen. Auch, ob eine gegebene Software tatsächlich aus einem zertifizierten Quellcode kompliliert worden ist, müßte überprüfbar sein.

Auch an die Informatikausbildung stellen sich neue Anforderungen. Ein Sicherheitsbewußtsein, die Einhaltung von RFCs, die Kommentierung von Code sollten im Curriculum stärker gewichtet werden. Bogk zufolge sind 95 Prozent aller Sicherheitsprobleme *Buffer Overflows*. Dabei handele es sich um triviale Fehler, deren Vermeidung in der Universität gelehrt werden müßte.

Zur Vermittlung der Prinzipien freier Software wäre es zudem an den öffentlichen Universitäten, mit gutem Beispiel voranzugehen. TCP/IP, BSD, der Linux-Kernel und zahlreiche weitere Software ist an Universitäten entwickelt worden. In den USA muß Software, die mit staatlichen Mitteln entwickelt wurde, allen zu Gute kommen. An deutschen Hochschulen gibt es diese Verpflichtung nicht. Durch die Zunahme der Drittmittelforschung gestaltet sich die Situation noch schwieriger. Auch hier gibt es -- nicht zuletzt im Interesse der Sicherheit -- ein Bedarf nach strukturellen Änderungen.

"Ich denke, daß diese ganze Auseinandersetzung um IT-Sicherheitsfragen ..., die momentan nur als Dichotomie gesehen wird von *Security by Obscurity*, die niemand überprüfen kann, versus Offenheit, die jeder überprüfen kann, eigentlich nur ein erster Schritt ist. Dieser erste Schritt wird dazu führen, daß wir eine Infrastruktur von Überprüfbarkeit und von vertrauenswürdigen Instanzen brauchen, die das auch beglaubigen kann, was wir im offenen Code lesen können."[596]

---

[595] Ruhmann, Wizards 7/1999

[596] Ruhmann, Wizards 7/1999

‘Betriebssystem’ für eine freiheitliche Gesellschaft

Wie gezeigt wurde, geht die offene Software-Kooperation bis auf die Frühzeit des Computers zurück -- und die freie Wissenskooperation bis auf die Anfangszeit der Menschheit überhaupt. Die Schließung des Wissens unter dem Konzept des ‘geistigen Eigentums’ besonders in der zweiten Hälfte des 20. Jahrhunderts drohte diese Tradition vergessen zu machen, wenn nicht einige Unverbesserliche vermeintlich unzeitgemäß an ihr festgehalten hätten. Aus dieser untergründigen Entwicklungslinie wurde schließlich ein gewaltiger Strom, der die Fundamente der proprietären Kathedralen zu unterhölen begann und die Schwächen der durch Geheimhaltungsvorschriften und Patente eingezäunten Wissensgenerierung und der so entstehenden intransparenten, gezielt inkompatiblen und mit FUD-Strategien (*Fear, Uncertainty, Doubt*[597]) gegen die Konkurrenz abgeschotteten Wissensprodukte aufgezeigte. Seit gut zwei Jahren konnte die freie Software einen kolossalen Zuwachs an *Mindshare* verzeichnen.

Etwas, wovon zuvor nur eine vergleichsweise kleine Gruppe von Beteiligten auch nur wußte, wurde zum Stoff für Titelgeschichten. Wirtschaftskreise, die überzeugt waren, daß es so etwas wie ein kostenloses Mittagessen nicht gebe, begannen sich dafür zu begeistern. Selbst auf dem Weltwirtschaftsforum in Davos gab es einen Workshop über Open Source-Software.[598] An den soziologischen, wirtschaftwissenschaftlichen, anthropologischen und informatischen Fakultäten werden Diplom- und Doktorarbeiten über das Phänomen geschrieben. Ministerien, Verwaltungen und Schulen beginnen, die Vorteile von freier Software zu erkennen und sie einzusetzen und zu fördern. Einiges bliebe noch zu wünschen, z.B. daß Ausschreibungen der öffentlichen Hand für Software, Systeme und verwandte Dienstleistungen technologieneutral spezifiziert würden, so daß nicht ohnehin nur Microsoft-Produkte in Frage kommen, sondern freie Software konkurrieren kann. “Das würde einerseits die kleineren und mittleren Unternehmen -- denn die sind es, die freie Software im Moment einsetzen und anbieten -- fördern und auf der anderen Seite auch eine Menge Geld für den Bundeshaushalt sparen.”[599]

Generell ist jedoch auch die öffentliche Politik den öffentlichen Wissensgütern gegenüber sehr aufgeschlossen. Ein bemerkenswertes Beispiel ist die Förderung des GNU Privacy Guard durch das Bundeswirtschaftsministerium. Ein anderes ist das Open Source-Zentrum *BerliOS*,[600] initiiert vom Forschungszentrum für Informationstechnik GmbH der Gesellschaft für Mathematik und Datenverarbeitung (GMD), das ab Herbst 2000 den Entwicklern und Anwendern Infrastruktur, Geld und Informationen zur Verfügung stellen will. BerliOS soll auch aktiv Anwenderbedürfnisse erforschen und Anstöße in Bereichen geben, in denen das Angebot freier Software heute noch ungenügend ist. Auch BerliOS erhält eine Anschubfinanzierung vom Bundeswirtschaftsministerium, das zudem noch eine Informationsbroschüre zu Open Source-Software für kleine und mittlere Unternehmen und die Verwaltung in Auftrag gegeben hat.

---

[597] "Mache den Leuten klar, daß 'da draußen' ein Krieg tobt, ein babylonisches Elend herrscht und nur der dominante Standard die beste Lösung bietet. Wer abweicht, wird mit Inkompatibilität bestraft. Wer mitmacht, kann nichts falsch machen." Schultz 12/1999

[598] Interview der c't mit EU-Kommissar für die Informationsgesellschaft Erkki Liikanen, http://www.ix.de/ct/00/09/058/

[599] Hetze, Fachgespräch 7/1999

[600] http://www.berlios.de

Neal Stephenson argumentiert überzeugend, daß Betriebsysteme Arbeitsumgebungen und Werkzeuge von Entwicklern und keine Konsumprodukte sind, und sie daher ihrer Natur nach frei zu seien haben.[601] Auch Peter Deutsch schrieb: “Software is essentially a public good”[602]

Der freie Geist breitet sich aus. Wo immer Probleme sich stellen, finden sich Leute, die eine ‘Open Source’-Lösung ausprobieren. Noch nah am Programmcode ist die Dokumentation von freier Software, die ebenfalls frei verbreitet und verändert werden können sollte,[603] doch auch andere Wissensformen, wie Bildungsmaterialien, literarische Texte, Musik und Enzyklopädien, werden heute von ihren Autoren als *Open Content* in den offenen dialogischen Wissensprozeß gestellt. Selbst an freier Hardware wird gearbeitet.[604] Das sicherlich spektakulärste Projekt war der Versuch dreier *Techies*, die 66 Satelliten des bankrotten Mobiltelefonie-Netzwerks Iridium von Motorola zum ersten öffentlichen Open Source-Netzwerk im Weltall zu machen.[605]

Bei aller Mächtigkeit, mit der sich die Wissensfreiheit Bahn bricht, ist sie doch immer gefährdet. Zu den aktuellen Bedrohung von außen gehört vor allem die Software-Patentierung, die, seit 20 Jahren in den USA gängige Praxis, im Herbst 2000 auch in Europa eingeführt werden soll.[606] Eine weiter klimatische Verschärfung stellen die Kampagnen der Software- und der Musikindustrie gegen das ‘Raubkopieren’ dar. Stallman vergleicht sie mit dem anhaltenden ‘Krieg gegen Drogen’ der US-Regierung. In den USA sitzen deshalb bereits ein Millionen Menschen im Gefängnis. Er führe zu Korruption und Verzerrung der Bürgerrechte. Der Krieg gegen das Kopieren könne noch schlimmere Folgen haben.

> “Think how much fear is going to be required to stop people from passing along copies of things on their computers. I hope you don't want to live in a world with that much fear. The Soviet Union tried to stop people from passing around copies of things, and they found a number of very interesting methods of preventing it. Today, the US government is proposing and enacting all of the same methods. It turns out that if you want to stop people from sharing copies of things, there are only certain methods that are applicable. It doesn't matter whether the motive is political censorship or simply enforcing the monopoly power for some business -- they use the same methods, and they make society monstrous in the same way.”[607]

Nicht die Software, sondern das Wissensregime und letztlich die Gesellschaft, in der wir leben wollen, steht für Stallman im Zentrum der Bewegung: “The core of the GNU project is the idea of free software as a social, ethical, political issue: what kind of society do we want to live in?”[608]

---

[601] Stephenson 1999

[602] Deutsch 1996

[603] vgl. Stallman 1997b

[604] z.B. http://www.openhardware.org; vgl. Perens 1999: 174

[605] *Save Our Sats* (http://www.saveiridium.com) wurde im März 2000 gestartet, doch hält Motorola offenbar an seinen Plänen fest, die milliardenschweren Satelliten Ende des Jahres kontrolliert über dem Pazifik zum Absturz zu bringen. (vgl. http://www.salon.com/tech/log/2000/03/23/save_iridium/print.html)

[606] s. die Eurolinux Petition for a Software Patent Free Europe, http://petition.eurolinux.org/

[607] Stallman, Wizards 7/1999

[608] Stallman, Wizards 7/1999

Doch auch von innene gibt es Bedrohungen. Die Monetarisierung des freien Austausches, die von solchen Projektbörsen wie Collab.net eingeleitet wird, könnte seinen Charakter grundlegend verändern. Natürlich soll ein Spitzwegscher ‘armer Programmierer’ mit Laptop auf dem Dachboden nicht zu einem Ideal erhoben werden. Das Hauptproblem sind vielmehr die Schließungen des Prozesses und seiner Resultat, die damit einhergehen.

Es ist deutlich geworden, daß freie Software nicht nur technische, sondern kulturelle, politische und ökonomische Implikationen hat. Sie, ebenso wie das Internet insgesamt, belegt, daß eine Selbstorganisation in der öffentlichen Domäne alternativ zur staatlichen Domäne und zur profitmaximierenden privatwirtschaftlichen Domäne machbar und in vieler Hinsicht äußerst attraktiv ist. Freie, quelloffene Software erlaubt Nutzern ein viel größeres Maß an Transparenz und Kontrolle über ihre Rechnerumwelt. Anwender sind nicht auf die Rolle von Konsumenten fertiger Produkte festgelegt, sondern sind Ko-Produzenten. Zusammen mit ihren bewährten Eigenschaften -- Flexibilität, Bildung, Innovation, Sicherheit, Stabilität, Kosteneinsparung -- ist freie Software somit ein attraktives Modell eines ‘Betriebssystems’ für eine freiheitliche Gesellschaft.

Was wir heute erleben, erinnert an die Vision von der ‘Noosphäre’, die der Jesuitenpater Pierre Teilhard de Chardin schon 1947 formulierte, als es nicht einmal eine Handvoll Computer auf der Welt gab und von ihrer Vernetzung noch keine Rede sein konnte. Bei der Noosphäre handelt es sich um eine planetare “Hülle aus denkender Substanz ... im Ausgang und oberhalb der Biosphäre”.[609] Das “radiophonische und televisionelle Nachrichtennetz [verbindet] uns alle schon jetzt in einer Art 'ätherischem' Mitbewußtsein. Vor allem denke ich hier aber an den fallenreichen Aufstieg dieser erstaunlichen Rechenmaschinen, die mit Hilfe von kombinierten Signalen in der Größenordnung von mehreren hundertausend in der Sekunde, ... die Denkgeschwindigkeit, also einen wesentlichen Faktor, in uns erhöhen.” (212 f.) Teilhard sieht darin “die Grundzüge einer besonderen Art von Super-Gehirn, das fähig ist, sich zu steigern, bis es irgendeinen Super-Bereich im Universum und im Denken meistert!" Die Noosphäre ist ein Gehirn aus Gehirnen. Alles geht vom Individuum aus, "doch alles vollendet sich oberhalb des Individuums." (224)

Man muß Teilhard nicht in die schwindelerregenden Höhen des Gottesproblems folgen, um zu sehen, daß das Internet viele der Qualitäten seiner Noosphäre verkörpert. Und ein Beispiel dafür, was die Verdichtung und die freie Assoziation von vielen, was ein “'ätherisches' Mitbewußtsein”, was das “Gehirn der Gehirne” zustande bringen kann, ist eben die freie Software.

Dieses Potential muß gepflegt und gegen die aggressiven Schließungsbestrebungen der globalen Rechteindustrie in Schutz genommen werden. “Die Freiheit des Wissens zu verteidigen, ist wahrscheinlich die wichtigste Aufgabe, die wir in der Zukunft vor uns haben.”[610] In dieser Überzeugung treffen sich Ökonom und Hacker.

---

[609] de Chardin 1966: 91

[610] Szyperski, Wizards 7/1999

Literatur

4C, Copy Protection Framework for DVD Audio, o.J.,
http://www.dvdcca.org/4centity/data/tech/dvd_audio_cp.pdf

Albini, Steve, The Problem With Music, o.J., http://www.negativland.com/albini.html

Alvestrand, Harald Tveit, “The Internet Standardisation Process”, in: T. Buland, H. Finne, S. Helmers, U. Hoffmann, J. Hofmann (Hrsg.), Managements and Network Technology, Proceedings from COST A3 Workshop in Trondheim, Nov. 22-24, 1995, WZB Paper 1996, S. 59-65

Baase, Sara, “IBM: PRODUCER or PREDATOR”, in: Reason, April 1974, pp. 4-10,
http://www-rohan.sdsu.edu/faculty/giftfire/ibm.html

Bachrach, Steven; R. Stephen Berry, Martin Blume, Thomas von Foerster, Alexander Fowler, Paul Ginsparg, Stephen Heller, Neil Kestner, Andrew Odlyzko, Ann Okerson, Ron Wigington, Anne Moffat: INTELLECTUAL PROPERTY: Who Should Own Scientific Papers? in: Science Volume 281, Number 5382 Issue of 4 Sep 1998, pp. 1459 - 1460
http://www.sciencemag.org/cgi/content/full/281/5382/1459

Baker, Steven, Net Worth. Desktop TCP/IP At Middle Age, in: UNIX Review, February 1998; http://www.performancecomputing.com/unixreview/backissu/9802/9802net.htm

Barlow, John Perry, “The Economy of Ideas. A framework for patents and copyrights in the Digital Age. (Everything you know about intellectual property is wrong.)”, in: Wired 2.03, März 1994, http://www.wired.com/wired/2.03/features/economy.ideas.html

Barlow, John Perry, “The Best of All Possible Worlds”, in: Communications of the ACM, 50th Anniversary Issue, 1996,
http://www.nettime.org/nettime.w3archive/199612/msg00011.html

Becker, Jürgen und Thomas Dreier (Hg.), Urheberrecht und digitale Technologie, UFITA-Schriftenreihe, Nomos Verlag, Baden-Baden 1994

Bell, Tom W., Fair Use Vs. Fared Use: The Impact of Automated Rights Management on Copyright's Fair Use Doctrine, in: 76 N. Carolina L. Rev. 557, S. 558-618, 1998,
http://www.tomwbell.com/writings/FullFared.html

Bell, Bruce, The Coming Storm, June 3, 2000,
http://eon.law.harvard.edu/openlaw/DVD/articles/comingstorm.html

Berners-Lee, Tim, Der Web-Report, Econ, München, 1999

Bezroukov, Nikolai, A Second Look at the Cathedral and the Bazaar, First Monday, 12/1999,
http://firstmonday.org/issues/issue4_12/bezroukov/

Borchers, Detlev, “Patente, die die Welt erschüttern”, in: ZD-Net Deutschland, August 1999; http://www.zdnet.de/kolumne/199908/mdb16_00-wc.html

Bortloff, Dr Nils, Legal Advisor, IFPI, “Internet Piracy - notice and take-down”, presentation at the WIPO Workshop on Service Provider Liability, Geneva, December 9 and 10, 1999, http://www.ifpi.org/antipiracy/notice_takedown.html und http://www.wipo.int/eng/meetings/1999/osp/doc/osp_lia3.doc

Bruce Perens, It's Time to Talk about Free Software Again, 1999a, http://www.perens.com/perens_com/Articles/ItsTimeToTalkAboutFreeSoftwareAgain.html

Buma/Stemra, Annual Report 1997

Burckhardt, Jacob, Die Kultur der Renaissance in Italien, Kröner, Stuttgart 1976

Busch, Christoph, Michael Arnold, Wolfgang Funk, “Schutz von Urheberrechten durch digitale Wasserzeichen”, in: Gerhard Banse und Christian J. Langenbach (Hrg.), Geistiges Eigentum und Copyright im multimedialen Zeitalter. Positionen, Probleme, Perspektiven. Eine fachübergreifende Bestandsaufnahme, Europäische Akademie, Bad Neuenahr-Ahrweiler 1999

Carter, Joseph, Why Debian Doesn't Include KDE, June 17th 2000, http://freshmeat.net/news/2000/06/17/961300740.html

Cerf, Vinton, How the Internet Came to Be, as told to Bernard Aboba, in: Bernard Aboba, The Online User's Encyclopedia, Addison-Wesley, November 1993, http://www.isoc.org/guest/zakon/Internet/History/How_the_Internet_came_to_Be

Chapman, D. Brent, “Majordomo: How I Manage 17 Mailing Lists Without Answering ‘-request’ Mail”, 1992 http://www.greatcircle.com/majordomo/majordomo.lisa6.ps.Z

Chardin, Pierre Teilhard de, Die Zukunft des Menschen, Olten 1966

CISAC, The Common Information System. A Digital Rights Architecture for the Information Age, Paris o.J.; http://www.worksnet.org/Web_cisac/cisac_communication.nsf/e9e109c34d8a0c2ec1256713004a879a/297492bec1062cb04125676a00401c38/$FILE/CIS_A.pdf

Clemins, Admiral Archie, Commander in Chief, U.S. Pacific Fleet, IT-21: The Path to Information Superiority, o.J., http://www.chips.navy.mil/chips/archives/97_jul/file1.htm

Cohen, Julie E., Some Reflections on Copyright Management Systems and Laws Designed to Protect Them, in: Berkeley Law Journal, 12 Berkeley Technology Law Journal 161, 1997, http://www.law.berkeley.edu:80/journals/btlj/articles/12_1/Cohen/html/text.html

Cohen, Julie E., A Right to Read Anonymously: A Closer Look at “Copyright Management” in Cybercpace, Connecticut Law Review, 981, 991, Summer 1996, http://cyber.law.harvard.edu/property/alternative/Cohen.html

Cusumano, Michael und Richard Selby, "How Microsoft Builds Software", *Communications of the ACM*, June 1997, 53-61

Czychowski, Christian, Ein Muster im Ringen um die Unabhängigkeit der Urheber -- Anhalt-Dessau-Wörlitz und seine Selbstverlagsunternehmen (1781-1785), in: forum historiae iuris, 19. August 1998, http://www.rewi.hu-berlin.de/FHI/98_08/czych_t.htm

Dell, Michael, Catherine Fredman, Direct From Dell: Strategies That Revolutionized an Industry, Harper Business, 1999

Demiralp, Emre, Linux in der akademischen Ausbildung, Linux-Focus, Oktober 1998, http://www.linuxfocus.org/Deutsch/October1998/article9.html

Deutsch, L. Peter, Licensing Alternatives for Freely Redistributable Software, 1996, ftp://ftp.cs.wisc.edu/ghost/papers

Dietrich, Oliver, “In den Tiefen des Kernel. Interview mit Linux-Entwickler Alan Cox”, in: c’t, 25/1999, S.34

Druey, Jean Nicolas, Information als Gegenstand des Rechts, Nomos, Baden-Baden 1995

DTLA (Digital Transmission Licensing Administrator), Policy Statements Regarding DTCP Adopters, January 26, 1999; http://www.dtcp.com/adopt.pdf

Dwyer, Michael, Linux has Window of Opportunity in China, in: Austrian Financial Review, June 30, 2000, http://www.afr.com.au/information/20000630/A42633-2000Jun29.html

Elkin-Koren, Niva, Law School, Haifa University, The Future of Public/Private Boundaries for Copyright in Cyberspace, Journal of Computer Mediated Communication, Vol. 2, No.2 Special Issue: "Emerging Law on the Electronic Frontier" (September 1996), http://www.ascusc.org/jcmc/vol2/issue2/elkin.html

Elkin-Koren, Niva, Public/Private and Copyright Reform in Cyberspace, 12:1 Berkeley Technology Law Journal, Spring 1997, http://eon.law.harvard.edu/property99/alternative/Elkin-Koren.html

Ellins, Julia, Copyright Law, Urherberrecht und ihre Harmonisierung in der Europäischen Gemeinschaft. Von den Anfängen bis zum Informationszeitalter, Duncker & Humblot, Berlin 1997

Ellins, Julia, Copyright law, Urheberrecht und ihre Harmonisierung in der Europäischen Gemeinschaft von den Anfängen bis ins Informationszeitalter, Duncker & Humblot, Berlin, 1997

Evers, Steffen, An Introduction to Open Source Software Development, Diplomarbeit an der TU-Berlin, Juli 2000

Fachgespräch Open Source-Software des Bundeswirtschaftsministeriums, am 15.7.1999, im Haus der Kulturen der Welt in Berlin

Flusser, Vilém, Ins Universum der technischen Bilder, Göttingen 1985

Fonda, Daren, Copyright crusader, in: The Boston Globe, 29. August 1999, http://www.boston.com/globe/magazine/8-29/featurestory1.shtml

Fritsch, Lothar, Die Geburt des Mikroprozessors, Saarbrücken, im April 1992, http://fsinfo.cs.uni-sb.de/~fritsch/Papers/PC/node9.html#SECTION00033000000000000000

Gabriel, Richard P. und William N. Joy, Sun Community Source License Principles, 20 January 1999, http://www.sun.com/981208/scsl/principles.html

Gartner Group Inc., The E-Market Maker Revolution, 1999, http://www.gartner.com/webletter/softlock/issue1/

Gehring, Robert, Freeware, Shareware und Public Domain. Geschichte, Begrifflichkeit, Urheberrecht und Haftung, Studienarbeit an der TU-Berlin, Juli 1996, http://ig.cs.tu-berlin.de/sa/043/index.html

Gehring,Robert, “Schneller, höher, weiter. Veranstaltungsbericht ‘Effizienter Staat 2000', in: Linux-Magazin 6/2000, S. 56-57

GEMA, Digitaltechnik und Urheberrecht im Bereich der Musik, Mai 1997, http://www.gema.de/service/digi.shtml

Gieseke, Michael, Der Buchdruck in der frühen Neuzeit, Suhrkamp, FfM 1991

Gilder, George, Metcalfe's Law and Legacy, in: Forbes ASAP, September 13, 1993, http://www.forbes.com/asap/gilder/telecosm4a.htm

Gomulkiewicz, Robert, “How Copyleft Uses License Rights to Succeed in the Open Source Software Revolution and the Implications for Article 2B”, in: Houston Law Review, Spring 1999, S. 180 ff., http://eon.law.harvard.edu/h2o/property/alternatives/gomulkiewicz.html

Götting, Horst-Peter (Hrsg.), Multimedia, Internet und Urheberrecht, Dresden University Press, Dresden & München 1998

Grassmuck, Volker, Open Source - Betriebssystem für eine freiheitliche Gesellschaft, in: Linux-Magazin 9/2000, S.54-61, http://waste.informatik.hu-berlin.de/Grassmuck/Texts/OSS-Tutzing-5-00.html.

Grassmuck, Volker und Christian Unverzagt, Das Müll-System. Eine Metarealistische Bestandsaufnahme, Suhrkamp, FfM 1991

Gravell, Tom, The Wizard of Watermarks, DU PONT Magazine, January/February 1990, pp. 4-6, http://ebbs.english.vt.edu/gravell/wizard/wizard.html

Grimm, Jacob und Wilhelm, Deutsches Wörterbuch, Bd. 1, A - Biermolke, Leipzig 1854, bei dtv, München 1984

Grimm, Jacob und Wilhelm, Deutsches Wörterbuch, Neubearbeitung, Berlin-Brandenburgische Akadmie der Wissenschaften und Akademie der Wissenschaften Göttingen, S. Mirzel Verlag, Stuttgart/Leipzig 1998

Guibault, Lucie, für Imprimatur, Legal SIG Workshop on contracts and copyright exemptions, "Contracts and Copyright Exemptions", Institute for Information Law, Amsterdam, Dezember 1997, http://www.imprimatur.alcs.co.uk/IMP_FTP/except.pdf

Gulbin, Jürgen und Karl Obermayr, Unix System V.4. Begriffe, Konzepte, Kommandos, Schnittstellen, Springer, Berlin, Heidelberg, New York, 1995

Halbert, Debora J., Weaving Webs of Ownership: Intellectual Property in an Information Age, Dissertation Draft, Hawaii University 1998, http://www.soc.hawaii.edu/~future/dissertation/TOC.html

Hardin, Garrett, "The Tragedy of the Commons", in: Science, 162 (1968), pp. 1243-1248, http://dieoff.com/page95.htm

Hardin, Garrett, Ethical Implications of Carrying Capacity, 1977, http://dieoff.com/page96.htm

Harsh, B., "New computer operating system takes India by storm", India Abroad News Service, auf: Corporate Watch, Globalization and Microsoft, Feb 9 1999, http://www.igc.org/trac/feature/microsoft/globalization/india.html

Hauben, Michael und Ronda Hauben, Netizens: On the History and Impact of Usenet and the Internet, 6/12/1996, http://www.columbia.edu/~hauben/netbook/

Hautsch, Gert, Zur Entwicklung der Medienwirtschaft in Deutschland 1997/98, 9. August 1998, http://www.igmedien.de/publikationen/m/1998/08-09/10.html

Helmers, Sabine und Kai Seidler, Linux: Cooperative Software Development and Internet, Proceedings of the First Dutch International Symposium on Linux - December 1994 at Amsterdam, Pages 56-59, State University of Gronigen, 1995, http://duplox.wz-berlin.de/docs/linux/index.html

Hetze, Sebastian, Wirtschaftliche Aspekte von Freier Software und OpenSource, 1999, http://mikro.org/Events/OS/ref-texte/hetze.html

Hetze, Sebastian, Dirk Hohndel, Martin Müller und Olaf Kirch, Linux Anwenderhandbuch. Leitfaden für die Systemverwaltung, LunetIX, Berlin 1997 und online: http://www1.lunetix.de/LHB//

Hitachi, Ltd., Intel Corporation, Matsushita Electric Industrial, Co., Ltd., Sony Corporation, Toshiba Corporation, Digital Transmission Content Protection Specification Volume 1 (Informational Version), Revision 1.0, April 12, 1999; http://www.dtcp.com/dtcp_spec1.pdf

Hitachi, Ltd., Intel Corporation, Matsushita Electric Industrial, Co., Ltd., Sony Corporation, Toshiba Corporation, 5C Digital Transmission Content Protection White Paper, Revision 1.0, July 14, 1998; http://www.dtcp.com/wp_spec.pdf

Horns, Axel H., DeCSS-Spruch - Modell für Europa?, in: Digest "Netz und Politik" 19, *, 2000, http://www.fitug.de/netpol/00/19.html#1

Intel Corporation, International Business Machines Corporation, Matsushita Electric Industrial Co., Ltd., Toshiba Corporation, “Content Protection System Architecture. A Comprehensive Framework for Content Protection”, February 17, 2000, Revision 0.81; http://www.dvdcca.org/4centity/data/tech/cpsa/cpsa081.pdf

Kahn, David, The Codebreakers; The Comprehensive History of Secret Communication from Ancient Times to the Internet, Scribner, New York 1996

Kant, Immanuel, Von der Unrechtsmäßigkeit des Büchernachdruckens, 1785; Nachdruck in UFITA 106, 1987, S. 137

Katz, Ronald S. und Janet Arnold Hart, Turning On and Turning Off. Can merely turning on a computer constitute copyright infringement? In the Ninth Circuit, yes, The Recorder, 1996, http://www.ipmag.com/katz.html

KBSt-Brief Nr. 2/2000, Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik, “Open Source Software in der Bundesverwaltung”, http://www.kbst.bund.de/papers/briefe/02-2000

Kelsey, John und Bruce Schneier, The Street Performer Protocol and Digital Copyrights, in: Firstmonday, Jahrg. 4, #6, 1999, http://www.firstmonday.dk/issues/issue4_6/kelsey/

Kingdon, Jim, (Cyclic), Free Software Business Models, December 1996, http://www.stromian.com/bizmod.html

Kirch, John, Microsoft Windows NT Server 4.0 versus Unix, March 1998, http://www.unix-vs-nt.org/kirch/

Kloke-Lesch, Adolf, “Funktionale Positionsbestimmung der Entwicklungspolitik”, in: Politik und Gesellschaft Online; International Politics and Society 3/1998 (Friedrich Ebert Stiftung), http://www.fes.de/ipg/ipg3_98/artkloke.html

Kohn, Alfie, Studies Find Reward Often No Motivator, Boston Globe, 1987-01-19, http://www.naggum.no/motivation.html

Köhntopp, Kristian, “RPS III: Wenn geistiges Eigentum zum Unwesen wird”, auf: "Netz und Politik"(NETPOL-Digest) 20, 10 Feb 2000, http://www.fitug.de/netpol/00/20.html#3

Kreile, Reinhold & Jürgen Becker, “Tarife für Multimedianutzung”, in: Rolf Moser & Andreas Scheuermann (Hrsg.), Handbuch der Musikwirtschaft, Joseph Keller Vrlg., Starnberg & München 1997b, S. 715-741

Kreile, Reinhold & Jürgen Becker, “Verwertungsgesellschaften”, in: Rolf Moser & Andreas Scheuermann (Hrsg.), Handbuch der Musikwirtschaft, Joseph Keller Vrlg., Starnberg & München 1997a, S. 621-647

Krempl, Stefan, “Mit URL-Blocker gegen MP3-Server”, in: Telepolis, 03.09.1999, http://www.heise.de/tp/deutsch/inhalt/te/5259/1.html

Krempl, Stefan, EU Commissioner: We need a sense of urgency, Interview with Erkki Liikanen, Telepolis, 17.04.2000, http://www.heise.de/tp/english/inhalt/te/8052/1.html

Kunze, Carol, Hotton Button Issue: Mass Market Licenses, 13. März 1997, http://www.2bguide.com/hbimmvc.html

Kuri, Jürgen, “Alice im Bücherland. Online-Buchläden: Die Speerspitze des E-Commerce?”, c't 19/99, S. 164, http://www.heise.de/ct/99/19/164/

Leonard, Andrew, The Richard Stallman Saga, Redux, Salon Magazine, 9/1998, http://www.salon.com/21st/feature/1998/09/11feature.html

Leonard, Andrew, “The Saint of Free Software. Maverick Richard Stallman Keeps his Faith -- and Gives Bill Gates the Finger”, in: Salonmagazine 8/1998, http://www.salon.com/21st/feature/1998/08/cov_31feature2.html

Lessig, Lawrence, Jefferson’s Nature, Draft 1, University of Virginia, Charlottesville, Va, November 19, 1998; http://cyber.law.harvard.edu/works/lessig/NatureD3.pdf

Lessig, Lawrence, Code and Other Laws of Cyberspace, Basic Books, New York 1999

Lessig, Lawrence, The Law of the Horse: What Cyberlaw Might Teach (final draft), 1999a http://cyber.law.harvard.edu/works/lessig/finalhls.pdf

Lettice, John, “French senators propose making open source compulsory”, The Register, 24/10/99, http://www.theregister.co.uk/991024-000005.html

Levy, Steven, Hackers. Heroes of the Computer Revolution (1984), Bantam Doubleday Dell Publishing, New York 1994

Lewis, Jeff, The Cathedral and the Bizarre, Mac-Opinion, 7. Juli 2000, http://www.macopinion.com/columns/macskeptic/00/07/07/

Loren, Lydia Pallas, The Purpose of Copyright, in: Open Spaces Quarterly, February 7, 2000; http://www.public.asu.edu/~dkarjala/publicdomain/Loren2-7-00.html

Loukides, Mike, Some Thoughts on the Sun Community Source License, O'Reilly Java Center, Thu, 09 Dec 1999, http://java.oreilly.com/news/loukides_0399.html

Loukides, Mike, The Chameleon and the Virus: More Thoughts on Java's Community License, o.J., O'Reilly-News, o.J., http://java.oreilly.com/news/java_license_0399.html

Love, Courtney, Courtney Love does the math. The controversial singer takes on record label profits, Napster and 'sucka VCs', Salon Magazine, June 14, 2000, http://www.salon.com/tech/feature/2000/06/14/love/

Lühr, Rüdiger, Diskussionsentwurf zum Urheberrechtsgesetz von Justizminister Schmidt-Jortzig vorgelegt, 29. Oktober 1998, http://www.igmedien.de/publikationen/m/1998/10/29.html

Malkin, G., Who's Who in the Internet. Biographies of IAB, IESG and IRSG Members, May 1992, http://www.ietf.org/rfc/rfc1336.txt?number=1336

Marx, Reiner und Gerhard Sauder (Hrsg. & Nachw.), Moritz contra Campe. Ein Streit zwischen Autor und Verleger im Jahr 1789, Werner Röhrig Verlga, St. Ingbert 1993

McKusick, Marshall Kirk, Twenty Years of Berkeley Unix. From AT&T-Owned to Freely Redistributable, in: C. DiBona, S. Ockman, M. Stone (Hrg.), Open Sources. Voices from the Open Source Revolution, O'Reilly Verlag, Sebastopol 1999, S. 31-46

Merten, Stefan, Gnu/Linux - Meilenstein auf dem Weg in die GPL-Gesellschaft?, 2000 opentheory.org/proj/gplgesellschaft/v0001.phtml

Metzger, Axel und Till Jaeger, "Open Source Software und deutsches Urheberrecht", in: GRUR Int., Okt. 99, S. 839-848, http://www.ifross.de/ifross_html/art1.html

Moglen, Eben, Anarchism Triumphant: Free Software and the Death of Copyright, First Monday, August 1999, http://old.law.columbia.edu/my_pubs/anarchism.html

Molnar, Daniel, Pop Muzik. Join the New Folkateers!, auf: Nettime-L, 29 Sep 1998

Münker, Reiner, Urheberrechtliche Zustimmungserfordernisse beim Digital Sampling, Peter Lang, Frankfurt/M, Berlin etc. 1995

Nadeau, Tom, The Case for Eminent Domain, first posted: 1 January 1998, http://www.vcnet.com/bms/features/domain.html

Negroponte, Nicholas, Being Digital, New York, Knopf, 1995

Neumann, A. Lin, Information Wants to Be Free - But This Is Ridiculous. We go on a shopping spree for pirate software in Asia, in: Wired 3.10 - Oct 1995, http://www.wired.com/wired/archive/3.10/piracy.html

Newitz, Annalee, “If code is free, why not me? Some open-source geeks are as open-minded about sex as they are about hacking” in: Salon Magazine, May 26, 2000; http://www.salon.com/tech/feature/2000/05/26/free_love/index.html

Newman, Nathan, From Microsoft Word to Microsoft World: How Microsoft is Building a Global Monopoly. A NetAction White Paper, 1997, http://www.netaction.org/msoft/world/

Nicholson, Bradley J., The Ghost In The Machine: MAI Systems Corp. v. Peak Computer, Inc. and the Problem of Copying in RAM, Berkeley Law Journal, 10/1995, http://www.law.berkeley.edu/journals/btlj/articles/10_1/Nicholson/html/text.html

O'Reilly, Tim, "Gated Source" Communities?, 7/5/2000, http://weblogs.oreillynet.com/tim/stories/storyReader$39

Oresme, Nicolas von, Traktat über Geldabwertungen, Kadmos Verlag, Berlin 1999

Patel, Prakesh, Open Source: Sizing the Linux Opportunity, WR Hambrecht + Co, 18. Mai 2000, http://www.wrhambrecht.com/research/coverage/opensource/ir/ir20000523.pdf

Perens, Bruce, The Open Source Definition, in: C. DiBona, S. Ockman, M. Stone (Hrg.), Open Sources. Voices from the Open Source Revolution, O’Reilly Verlag, Sebastopol 1999, S. 171-188

Petitcolas, Fabien A. P., Ross J. Anderson, Markus G. Kuhn. Attacks on copyright marking systems, in David Aucsmith (Ed), Information Hiding, Second International Workshop, IH'98, Portland, Oregon, U.S.A., April 15-17, 1998, Proceedings, LNCS 1525, Springer-Verlag, ISBN 3-540-65386-4, pp. 219-239. http://www.cl.cam.ac.uk/~fapp2/papers/ih98-attacks/

Petitcolas, Fabien A. P. und Ross J. Anderson, Evaluation of copyright marking systems. In proceedings of IEEE Multimedia Systems (ICMCS'99), vol. 1, pp. 574--579, 7--11 June 1999, Florence, Italy. http://www.cl.cam.ac.uk/~fapp2/papers/ieeemm99-evaluation/

Pfennig, Gerhard, “Die Funktionärsclique aus Frankfurt. Die Geschite der Wahrnehmung visueller Urheberrechte am Beispiel der VG Bild-Kunst”, in: Gerhard Pfennig & Michael Schwarz (Hrsg.), Die Zukunft der Bilder. Medienentwicklung und Recht - 25 Jahre VG Bild-Kunst, Steidl, Göttingen 1993, S. 12-29

Pfennig, Gerhard, Die Funktionärsclique aus Frankfurt. Die Geschichte der Warhnehmung visueller Urheberrechte am Beispiel der VG Bild-Kunst, in: Gerhard Pfennig, Michael Schwartz (Hg.), Die Zukunft der Bilder. Medienentwicklung und Recht -- 25 Jahre VG Bild-Kunst, Steidl, Göttingen 1993

Philipkoski, Kristen, The Student Jukebox Sting. The Internet may no longer be the Wild West. Seventy-one college students at Carnegie Mellon University (CMU) in Pittsburgh, Pennsylvania, have been disciplined for the illegal use of MP3 files on the University's intranet, Wirede News 4:20 p.m. Nov. 9, 1999 PST; http://www.wired.com/news/culture/0,1284,32444,00.html

Platt, Charles, Satellite Pirates, Wired 2.08 - Aug 1994,
http://www.wired.com/wired/archive/2.08/satellite.html

Plotz, David, Luke Skywalker Is Gay? Fan fiction is America's literature of obsession, posted Friday, April 14, 2000, auf Slate http://slate.msn.com/Features/fanfic/fanfic.asp

Pomian, Krzystof, Der Ursprung des Museums. Vom Sammeln, Wagenbach, Berlin 1998

Pomian, Krysztov 1998: Der Ursprung des Museums. Vom Sammeln, Wagenbach, (Paris 1986-88), Berlin (1988)

Powell, Dennis E., Judgement Day for the GPL?, 26. Juni 2000,
http://www.linuxplanet.com/linuxplanet/reports/2000/1/

Powell, Dennis E., Lawyers, Guns, and Money. KDE and Corel: Working in the Real World, LinuxPlanet, 14. Juni 2000a, http://www.linuxplanet.com/linuxplanet/reports/1960/1/

Raymond, Eric, The Cathedral and the Bazaar, Januar 1998,
http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar.html

Raymond, Eric S., “The Cathedral and the Bazaar”,
http://sagan.earthspace.net/~esr/writings/cathedral-bazaar/

Rice, Dan und Robert Pecot, Microprocessor History, o.J.,
http://members.tripod.com/nikkicox/

Rifkin, Jeremy, Vorabdruck aus: Access. Das Verschwinden des Eigentums. Wenn alles im Leben zur bezahlten Ware wird, Campus FfM 2000, in: FAZ, 12.8.2000

Rosenberg, Scott, Why the music industry has nothing to celebrate. Napster's shutdown will only cause a thousand alternatives to bloom, Salon Magazine, July 27, 2000
http://www.salon.com/tech/col/rose/2000/07/27/napster_shutdown/print.html

Rost, Martin, “Vorschläge zur Entwicklung einer wissenschaftlichen Diskurs-Markup-Language” in: Heibach, Christiane/ Bollmann, Stefan (Hrsg.), 1996: Kursbuch Internet - Anschlüsse an Wirtschaft und Politik, Wissenschaft und Kultur: Bollmann-Verlag,
http://www.netzservice.de/Home/maro/home/mr_dml.html

Rötzer, Florian, Napster-ähnliche Tauschbörse für die Wissenschaft. Besonders die Zusammenführung der Erläuterungen zum menschlichen Genom von verschiedenen Websites lässt neue Programme notwendig werden, in: Telepolis, 17.08.2000,
http://www.heise.de/tp/deutsch/inhalt/lis/8561/1.htm

Rump, Dipl.-Inf. Niels, Protecting Intellectual Property Rights using the Multimedia Protection Protocoll (MMP), Fraunhofer IIS, Erlangen, 19 August 1997 (nicht länger online)

Saacke, Astrid, “Schutzgegenstand und Rechtsinhaberschaft bei Multimediaprodukten - Grundfragen des Rechtsschutzes und der Lizenzierung”, in: Horst-Peter Götting (Hrsg.), Multimedia, Internet und Urheberrecht, Dresden University Press, 1998, S. 19-45

Samuelson, Pamela, Legally Speaking: Does Information Really Want To Be Licensed?, in: Communications of the ACM, September 1998;
http://www.press.umich.edu/jep/04-03/samuelson.html

Samuelson, Pamela, “Is information property? (Legally Speaking)”, in: Communications of the ACM, March 1991 v34 n3 p15(4),
http://www.ifla.org/documents/infopol/copyright/samp6.txt

Sanders, Glenn und Wade Roush, Cracking the Bullet: Hackers Decrypt PDF Version of Stephen King eBook, eBookNet.com, 23. März 2000,
http://www.ebooknet.com/story.jsp?id=1671

Scanlon, Jessie, Open Source: Power Exchange, Wired, Nov 1999,
http://www.wired.com/wired/archive/7.11/mustread.html?pg=2

Schreiner, Klaus, “Bücher, Bibliotheken und ‘gemeiner Nutzen’ im Spätmittelalter und in der frühen Neuzeit” in: Bibliothek und Wissenschaft 9, 1975: 216-249

Schultz, Pit, “Was ist digitaler Kapitalismus?”, in: de:bug, 12/99

Schulze, Marcel, Materialien zum Urheberrechtsgesetz. Texte - Begriffe - Begründungen, VCH, Weinheim, New York usw. 1993

Schulze, Martin, “Debian GNU”, in: FIFF-Kommunikation 3/1999, S. 27-33

Schulzki-Haddouti, Christiane, “Dialog mit Hindernissen, Bundeskriminalamt will Berühungsängste der Provider abbauen”, in: Telepolis, 17.02.2000,
http://www.heise.de/tp/deutsch/inhalt/te/5806/1.html

Seiferth, C. Justin, Maj USAF, Open Source and these United States, A Research Report Submitted to the Faculty, In Partial Fulfillment of the Graduation Requirements, Maxwell Air Force Base, Alabama, April 1999
http://skyscraper.fortunecity.com/mondo/841/documents/99-184.html

Siepmann, Jürgen, “Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software”, in: JurPC. Internet-Zeitschrift für Rechtsinformatik, 1999
http://www.jurpc.de/aufsatz/19990163.htm

Sollfrank, Cornelia, Woman Hackers, Rotterdam 1999, http://www.obn.org/hackers/text.htm

Sommerfeld, Bernd, “Wie es begann... Eine kurze Geschichte von Linux und “Open Source’”, 1999, http://www.de.uu.net/shop/jfl/linux/BSo7.html

Spinner, Helmut F., aus: Ettersburger Gespräche: Zur Ohnmacht verdammt? -- Der Staat in der Informationsgesellschaft, hrg. vom Thüringer Ministerium für Bundesangelegenheiten i.d. Staatskanzlei, Weimar, Feb 1998a

Spinner, Helmut F., Die Wissensordnung. Ein Leitkonzept für die dritte Grundordnung des Informationszeitalters, Leske + Budrich, Opladen, 1994

Spinner, Helmut F., Die Architektur der Informationsgesellschaft, Philo, Bodenheim 1998

Stallman, Richard, The Right Way to Tax DAT, in: Wired 1992, http://www.gnu.org/philosophy/dat.html

Stallman, Richard, Why “Free Software”' is better than “Open Source”, 1999b, http://www.gnu.org/philosophy/free-software-for-freedom.html

Stallman, Richard, Why you shouldn't use the Library GPL for your next library, February 1999a, http://www.gnu.org/philosophy/why-not-lgpl.html

Stallman, Richard, The GNU Manifesto, 1985, http://www.gnu.org/gnu/manifesto.html

Stallman, Richard, Free Software and Free Manuals, 1997b, http://www.fsf.org/philosophy/free-doc.html

Stallman, Richard, Why Software Should Not Have Owners, 1994, http://www.gnu.org/philosophy/why-free.html

Stallman, Richard, The Right to Read, in: Communications of the ACM, Vol. 20, No. 2, Februar 1997, http://www.gnu.org/philosophy/right-to-read.html

Stallman, Richard, The GNU Operating System and the Free Software Movement, in: C. DiBona, S. Ockman, M. Stone (Hrg.), Open Sources. Voices from the Open Source Revolution, O’Reilly Verlag, Sebastopol 1999, S. 53-70

Stallman, Richard, Copyleft: Pragmatic Idealism, 1998, http://www.gnu.org/philosophy/pragmatic.html

Starrett, Robert A., Copying Music to CD: The Right, the Wrong, and the Law, February 1998, http://www.cdpage.com/Audio_Compact_Disc/rightwrong.html

Stefik, Mark, Special Report: Trusted Systems. Devices that enforce machine-readable rights to use the work of a musician or author may create secure ways to publish over the Internet, Scientific American 3/1997b, 276(3):78-81; http://www.sciam.com/0397issue/0397stefik.html

Stefik, M. J., Letting loose the light: igniting commerce in electronic publication. In: Stefik, M., ed. Internet Dreams: Archetypes, Myths, and Metaphors, MIT Press, Cambridge, MA, 1996;

http://www.parc.xerox.com/istl/projects/uir/pubs/pdf/UIR-R-1996-10-Stefik-InternetCommerce-IgnitingDre

Stefik, Mark, Shifting the Possible: How Digital Property Rights Challenge Us to Rethink Digital Publishing, 12 Berkeley Technology Law Journal, pp. 137-159 1997a; http://www.law.berkeley.edu/journals/btlj/articles/12_1/Stefik/html/reader.html

Stephenson, Neal, In the Beginning was the Command Line, 1999, http://www.cryptonomicon.com/beginning.html

Steurer, Reinhard, "Die schwierige Psychologie der Umweltpolitik: Das Beispiel Österreich" (vorläufige Fassung), in: Politik und Gesellschaft Online, International Politics and Society 4/1999, http://www.fes.de/ipg/ipg4_99/artsteur.htm

Straub, Craig, Living in a World of Limits. An interview with noted biologist Garrett Hardin, in: The Social Contract, TSC Press, Fall, 1997, http://www.lrainc.com/swtaboo/stalkers/tsc_hard.html

Strauss, Neil, "Hong Kong Film: Exit the Dragon?", in: New York Times, August 1, 1998, http://www.geocities.com/Athens/Forum/2496/vcd-hkfilm.txt

Tiemann, Michael, Future of Cygnus Solutions: An Entrepreneur's Account, in: C. DiBona, S. Ockman, M. Stone (Hrg.), Open Sources. Voices from the Open Source Revolution, O'Reilly Verlag, Sebastopol 1999, S. 71-89

Tushnet, Rebecca, Using Law and Identity to Script Cultural Production: Legal Fictions: Copyright, Fan Fiction, and a New Common Law, Loyola of Los Angeles Entertainment Law Journal, 17 Loy. L.A. Ent. L.J. 651, 1997, http://cyber.law.harvard.edu/property/respect/fanfiction.html

Ulmer, Eugen, Urheber- und Verlagsrecht, 3., neubearb. Aufl., Springer, Berlin 1980

Valloppillil, Vinod, Open Source Software. A (New?) Development Methodology (aka "The Halloween Document"), Microsoft Confidential (mit Kommentaren versehen von Eric Raymond), Aug 11, 1998 -- v1.00., http://www.tuxedo.org/~esr/halloween.html

Vanpelt, Lauren, Mickey Mouse -- A Truely Public Character, Frühjahr 1999, http://www.public.asu.edu/~dkarjala/publicdomain/Vanpelt-s99.html

Vermeer, Martin, Linux and Ethnodiversity, Jan 21st 1998, http://linuxtoday.com/stories/2465.html

Verzola, Roberto, Cyberlords: The Rentier Class of the Information Sector, Philippines 1998, posted on Nettime, http://www.desk.nl/~nettime/

Vobruba, G., Strukturwandel der Sozialpolitik - Lohnarbeitszentrierte Sozialpolitik und soziale Grundsicherung, Suhrkamp, Frankfurt/M. 1990

Walsh, Mary Williams, “Windows won't compute into ancient Icelandic language”, in: Seattle Times, June 30, 1998, http://archives.seattletimes.nwsource.com/cgi-bin/texis/web/vortex/display?slug=icel&date=19980630

Weber, Max, Schriften zur Soziologie, Reclam, Stuttgart 1995

Wizards of OS. Offene Quellen und Freie Software, Konferenz, am 16.-17.7.1999, im Haus der Kulturen der Welt in Berlin, http://www.mikro.org/wos

Zawinski, Jamie, Resignation and Postmortem, March 99, http://www.jwz.org/gruntle/nomo.html

Zawinski, Jamie, Fear and loathing on the merger trail, 23-Nov-98, http://www.mozilla.org/fear.html

Zentralinstitut für Sprachwissenschaft, Berlin, u. Ltg. v. Wolfgang Pfeifer, Etmologisches Wörterbuch des Deutschen, dtv, München 1995

Zingel, Wolfgang-Peter, (Südasien-Institut der Universität Heidelberg, Abteilung Internationale Wirtschafts- und Entwicklungspolitik), “Bodenrecht in Indien”, Eine leicht gekürzte Fassung erschien in : “entwicklung + ländlicher raum. Beiträge zur internationalen Zusammenarbeit”. Frankfurt: DLG. 29(1995)6. pp. 7-10, http://www.sai.uni-heidelberg.de/intwep/zingel/bodenr95.htm

Zingel, Wolfgang-Peter, Genug Nahrung für eine Milliarde Inder? in: Werner Draguhn (Hrsg.): Indien 1999: Politik, Wirtschaft, Gesellschaft. Hamburg: Institut für Asienkunde. 1999; http://www.sai.uni-heidelberg.de/intwep/zingel/jbindi99.htm