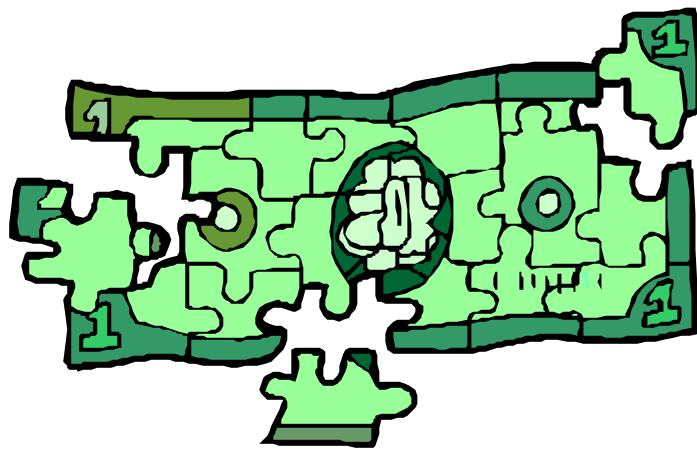


OPEN SOURCE SUCCESS



Referatsdokumentation zum Vortrag vom 15.12.03

Geschrieben von Christian Friedrich, Matthias Schulz und Ali Beyaz

Inhaltsübersicht

1. Einleitung
2. Geschichte von „Open Source“
3. Unterscheidung der Softwaretypen
4. Diskussionen der Vor- und Nachteile von „Open Source“
 - a. Diskussionspunkt: Support
 - b. Diskussionspunkt: Kosten
 - c. Diskussionspunkt: Abhängigkeit
 - d. Diskussionspunkt: Rechtssicherheit
5. Fazit

Einleitung

In den letzten 20 Jahren haben sich Computer von großen unhandlichen Kisten, die nur von gut ausgebildeten Leuten und einigen Freaks bedient werden konnten, zu praktischen Alltagsgeräten entwickelt, die in kaum einem Haushalt fehlen. Auch in vielen Unternehmen ist der Computer elementarer Bestandteil der Arbeit und somit gar nicht mehr wegzudenken.

Bei näherer Betrachtung dieser Entwicklung fällt einem allerdings folgendes auf: in den letzten Jahren wurde „Computer benutzen“ immer mehr zu „Microsoft benutzen“. Microsoft hat mit seinem Betriebssystem „Windows“ ein absolutes Monopol und jegliche Alternativen wurden vollständig oder in kleinere Nischen zurückgedrängt. Hinzu kommt noch, dass Microsoft mit seinem „Office“ das Monopol bei Bürosoftware hält. Diese Software braucht im Prinzip jede Firma, um die alltäglichen Arbeiten zu erledigen.

Dazu brauchen die Firmen natürlich noch andere Software, die für ihre speziellen Anforderungen geeignet sind, wie z.B. Buchhaltungssoftware, CAD/CAM, Grafikprogramme etc. Diese Programme werden dann bei speziellen Anbietern gekauft. Da es sich hier teilweise um sehr komplexe Computerprogramme handelt, die zudem nur in sehr begrenzter Stückzahl hergestellt werden, sind diese „Proprietäre Software“ natürlich auch dementsprechend kostspielig.

Dies ist der jetzige Zustand und so wie es aussieht, funktioniert dieses System soweit das alle beteiligten Firmen ihr Geld verdienen. Allerdings ist in den letzten Jahren und Monaten immer wieder von „Open Source“ und „Freier Software“ im Zusammenhang mit (zum Teil nicht unbedeutenden) Firmen die Rede.

Was bewegt immer mehr Firmen, von einer scheinbar funktionierenden Lösung abzulassen und auf „Freie Software“ Software umzusteigen? Woher kommt eigentlich diese Software? Was ist eigentlich „Open Source“? Mit all diesen Fragen werden wir uns beschäftigen. Dazu werden wir zuerst die Geschichte von „Open Source“ näher beleuchten, bevor wir dann die verschiedenen Typen von Software unterscheiden werden. Danach kommt dann der Hauptteil des Vortrages, wo wir (vermeintliche) Nachteile von „Open Source“ gegenüber von „Proprietärer Software“ diskutieren. Dazu werden wir zu den vier Punkten Support, Kosten, Abhängigkeit und Rechtssicherheit einige Punkte darstellen, wo „Open Source Software“ noch Nachteile hat und diese dann im Anschluss versuchen zu entkräften. Zu guter Letzt werden wir noch eine kurze Zusammenfassung geben und versuchen ein Fazit zu ziehen.

2. Geschichte von „Open Source“

Obwohl „Open Source“ erst in den letzten Jahren richtig „in Mode“ gekommen ist, gibt es diese Art des Programmierens annähernd so lange wie man Software für Computer entwickelt. Denn gerade am Anfang der Entwicklung von Computern waren die Universitäten führend in der Entwicklung von neuen Programmen. Und die meisten dieser Programme konnten von anderen Mitgliedern von Universitäten und sogar anderen Freiwilligen eingesehen und verbessert werden [2-1].

Das man für Software bezahlen muss und diese dann auch nicht mehr verändern kann, entwickelte sich erst mit der Entstehung von den unzähligen Softwarefirmen, die ihre Produkte möglichst gewinnbringend verkaufen mussten.

Open Source - wie wir es heute kennen - ist aus den beiden oben genannten Gründen entstanden: zum einem aus der Vorstellung, dass jeder das Recht haben sollte, seine eigenen Ideen in ein vorhandenes Stück Software einzubringen oder aber man seine Software Anderen zur Verfügung stellt, damit diese dann ihre Ideen verwirklichen können. Zum anderen aber spielen auch wirtschaftliche Aspekte eine Rolle. Man wollte nicht auf einmal viel Geld für Software bezahlen müssen, die vorher fast umsonst war.

Am Anfang war Unix ...

Der Grund, dass Open Source heutzutage in aller Munde ist, ist im Prinzip das Betriebssystem UNIX. Es handelt sich hier um den Nachfolger des Betriebssystem MULTICS (**M**ultiplexed **I**nformation and **C**omputing **S**ervice). MULTICS wurde entwickelt, um mehreren Benutzern gleichzeitig Zugang zu einem Computer zu ermöglichen und ihre Programme möglichst gleichmäßig auf die vorhandene CPU zu verteilen (Timesharing-Systeme). [2-2]

Dieses MULTIX wurde von den Bell Telephone Laboratories zusammen mit dem damaligen Computer-Hersteller General Electric und dem MIT entwickelt. Es wurde zwar eine lauffähige Version fertig gestellt, aber man war bei den Bell Telephone Laboratories nicht zufrieden mit den Resultaten, und so stieg man aus diesem Konsortium aus. Da jetzt einige Mitarbeiter der Bell Labs ohne „geeignete Programmierumgebung“ waren, begannen sie 1969 mit der Entwicklung eines eigenen Betriebssystems. [2-3]

Diese Mitarbeiter waren im wesentlichen Ken Thompson und Dennis Ritchie [2-4]. Sie entwickelten dann für einen Großrechner ihr eigenes Betriebssystem und nannten es UNIX (**U**niplexed **I**nformation and **C**omputing **S**ervice - in Anlehnung an MULTICS, das CS am Ende

wurde durch ein X ersetzt). Dieses System funktionierte auch, erzielte aber seinen großen Durchbruch erst, als es 1973 von Assembler in die portable Programmiersprache C umgeschrieben wurde. Ab 1977 gab es dann auch Versionen von UNIX, die nicht nur auf PDP Workstations liefen [2-3].

UNIX wird zum Universitäts-Standard

Der Grund für die weite Verbreitung von UNIX lag aber nicht (nur) an seinen herausragenden technischen Eigenschaften. Ein ganz wichtiger Aspekt ist, dass es (für damalige Verhältnisse) sehr günstige Lizenzierungskosten hatte. Der Grund dafür lag im „Consent Decree“ von 1956 [2-5]. Aufgrund einiger Kartellrechtsklagen durfte AT&T keine Gewinne bei Geschäften außerhalb des Telekommunikationssektors machen. Da die „Bell Laboratories“ eine Tochtergesellschaft von AT&T ist und Computerprogramme nicht direkt im Telekommunikationssektor liegen, wurde UNIX gegen eine (sehr) geringe Lizenzgebühr an staatliche Einrichtungen und Universitäten weitergegeben [2-6].

AT&T stellte allerdings keinen Support und keine Bugfixes zur Verfügung. Die Folge war eine rege Entwicklungstätigkeit im Umfeld der Universitäten. Besonders die Universität von Berkeley tat sich hier hervor. Ein Grund dafür ist sicherlich, dass Ken Thompson hier 1976 Gastprofessor war [2-7]. Hier wurden mit den Berkeley-Software-Distributionen wichtige Impulse zur Weiterentwicklung von UNIX gegeben [2-8]. Zusätzlich zu den universitären UNIX-Derivaten gab es ab 1982 noch kommerzielle Varianten, darunter Versionen von so bekannten Firmen wie Microsoft (XENIX) und IBM (AIX).

Das Ende vom „freien Unix“

Die freie Entwicklung an den Universitäten endete im Jahre 1984. In diesem Jahr wurde AT&T in einem weiteren Kartellrechtsprozess in 26 Einzelgesellschaften zerschlagen. Diese Einzelgesellschaften konnten nun allerdings als Mitbewerber im Softwaremarkt auftreten, und so dauerte es nicht lange, bis es eine eigene UNIX-Distribution gab und die Universitäten endgültig in ihrem Umgang mit UNIX stark eingeschränkt wurden [2-6].

Durch die immer rigideren Restriktionen für die UNIX-Lizenzierung und den Umstand, dass immer mehr Software in Form von Binärdateien und nicht in Form des Quellcodes weitergegeben wurde, regte sich langsam Widerstand bei den Programmierern, die es gewohnt waren, dass Programme frei verändert werden können.

Richard Stallman sucht nach Alternativen.

Richard Stallman war der erste der daraufhin aktiv wurde und 1983 sein „GNU-Projekt“ (Gnu is not UNIX) startete. Dies ist die deutsche Übersetzung seines „Initial Statement“ aus einer Newsgroup, in dem er am 27.09.1983 seinen Beschluss verkündete, GNU zu programmieren [2-9].

```
From CSvax:pur-ee:inuxc!ixn5c!ihnp4!houxm!mhuxi!eagle!mit-vax!mit-eddie!RMS@MIT-OZ
From: RMS%MIT-OZ@mit-eddie
Newsgroups: net.unix-wizards,net.usoft
Subject: new UNIX implementation
Date: Tue, 27-Sep-83 12:35:59 EST
Organization: MIT AI Lab, Cambridge, MA
```

Unix befreien!

An diesem Erntedankfest werde ich damit beginnen, ein komplettes Unix-kompatibles Softwaresystem mit dem Namen GNU (für Gnu ist Nicht Unix) zu schreiben und es freizugeben für jeden, dem es von Nutzen sein kann. Beiträge von Zeit, Geld, Programmen und Ausrüstung werden dringend benötigt.

Für den Anfang wird GNU ein Kernel und alle für das Schreiben und Laufenlassen von C-Programmen benötigten Utilities sein: Editor, Shell, C-Compiler, Linker, Assembler und ein paar andere Dinge. Danach werden wir einen Textformatierer, ein YACC, ein Empire-Spiel, eine Tabellenkalkulation und hunderte anderer Dinge hinzufügen. Wir hoffen, möglicherweise alles nützliche anzubieten, was normalerweise mit einem Unix-System daherkommt und alles andere Nützliche, darunter Online- und gedruckte Dokumentation.

GNU wird Unix-Programme ausführen können, aber nicht mit Unix identisch sein. Wir werden alle angenehmen Verbesserungen durchführen, basierend auf unserer Erfahrung mit anderen Betriebssystemen.

...

Wer bin ich?

Ich bin Richard Stallman, Erfinder des ursprünglichen, oft imitierten EMACS Editors, jetzt am Labor für Künstliche Intelligenz des MIT. Ich habe extensiv an Compilern, Editoren, Debuggern, Kommandointerpretern, dem Inkompatiblen Timesharing System und dem Lisp-Maschinen Betriebssystem gearbeitet.

...

Warum ich GNU schreiben muß

Ich denke, daß die Goldene Regel es erfordert, daß ich, wenn ich ein Programm mag, es mit anderen Leuten, die es mögen, teilen muß. Ich kann nicht guten Gewissens eine Nichtveröffentlichungserklärung oder eine Softwarelizenzklärung unterschreiben.

Damit ich weiterhin Computer benutzen kann ohne meine Prinzipien zu verletzen, habe ich entschieden, einen ausreichenden Körper an freier Software zusammenzustellen, so daß ich in der Lage sein werde, ohne jede nicht freie Software zurechtzukommen.

...

Stallman wollte keine Software, die ihn beim Benutzen einschränkte. Er kam zu dem Schluss, dass man als erstes ein freies Betriebssystem brauchen würde, um darauf freie Software auszuführen. Dieses Betriebssystem sollte allerdings nicht nur die notwendigsten Fragmente enthalten, sondern sollte alle Teile mit an Board haben, die man braucht um mit ihm vernünftig

zu arbeiten. Daher wurde erstmal begonnen, wichtige Programme zu schreiben, die essentiell für ein Betriebssystem sind, wie z.B. ein C-Compiler (gcc) oder eine Shell (bash).

Die GPL und die Free Software Foundation

Zusätzlich zur Schaffung des Betriebssystems war es für Stallman ebenfalls wichtig, eine rechtliche Grundlage dafür zu schaffen, dass freie Software auch frei bleibt und nicht von Anbietern proprietärer Software für ihre Zwecke missbraucht werden kann. Dazu entwickelte er das „Copyleft“. Die „zentrale Idee vom Copyleft ist, dass wir jedem die Erlaubnis geben, das Programm laufen zu lassen, es zu kopieren, zu modifizieren und modifizierte Versionen zu vertreiben -- aber nicht die Erlaubnis, eigene Begrenzungen hinzuzufügen“ [2-10]. Diese Richtlinien, wie Software weitergegeben werden darf / soll, wurde in der GNU General Public License (GNU GPL) festgelegt.

Außerdem gründete Stallman die „Free Software Foundation“ (FSF). Diese war ein „steuerfreier Wohlfahrtsverband für die Entwicklung freier Software“ [2-10]. Die FSF nahm Spenden entgegen (sowohl finanzielle als auch Sachspenden wie Computer) und verdiente weiterhin Geld damit, die Programme an Leute zu verschicken, die keinen Internetanschluss hatten und sich die Programme nicht kopieren konnten. Außerdem stellten sie noch Handbücher für die entsprechende Software bereit. Mit diesem Geld wurden dann Programmierer bezahlt, die noch fehlende Bestandteile für das GNU-Projekt erstellen sollten.

GNU-Software setzt sich langsam durch

Da das GNU-System ja kompatibel zu UNIX sein sollte, liefen die meisten Programme auch unter UNIX – teilweise besser als die originalen Pendanten [2-10]. Aus diesem Grund wurden die Programme von einer großen Gemeinschaft gepflegt und beständig verbessert.

Allerdings geriet durch diese vielen Verbesserungen ein Hauptziel des GNU-Projektes in den Hintergrund: die Entwicklung des Kernels. Erst 1990 wurde mit der Arbeit am Kernel HURD begonnen. Zum einen durch technische Schwierigkeiten, zum anderen aber auch durch die Tatsache, dass zu viele Ressourcen mit der Weiterentwicklung von anderen Projekten beschäftigt waren, verzögerten die Entwicklung des Kernels immer weiter [2-6].

Linus Torvalds entwickelt das fehlende Teil

An dieser Stelle kam der finnische Student Linus Torvalds ins Spiel. Er nahm sich das experimentelle Betriebssystem MINIX (ein vom holländischen Professor Andrew S. Tanenbaum als Lehrstück konzipiertes Betriebssystem) zum Vorbild und programmierte seinen eigenen Kernel - Linux! [2-11]

Da der Kernel HURD des GNU-Projekts auf absehbare Zeit nicht fertig zu werden schien, sprangen viele Entwickler sofort mit auf den Zug und entwickelten fortan für Linux. An und für sich kann man natürlich mit einem Kernel allein nichts anfangen, aber zusammen mit den zahlreichen Tools, die im Rahmen des GNU-Projektes entwickelt wurden, wurde daraus das erste komplett freie Betriebssystem.

Selbstverständlich funktionierten alle GNU-Programme nicht auf Anhieb und es musste noch viel Anpassungsarbeit geleistet werden, aber am Ende standen die ersten Versionen des Betriebssystems das heute alle nur als Linux kennen. Dadurch, dass (fast) alle Welt dieses Betriebssystem nur Linux nennt, erhält Richard Stallman und sein GNU-Projekt nicht die Wertschätzung, die es eigentlich verdient hätte.

Ideelle Probleme bei GNU und Linux

So ist der Anteil des eigentlichen Kernels am Umfang der Gesamtsoftware einer Linux-Distribution ziemlich gering im Vergleich zu dem Anteil der GNU-Software [2-12]. Aus diesem Grund spricht Stallman immer vom GNU/Linux, was für viele Leute aber zu umständlich ist, so dass sie der Einfachheit halber immer nur Linux sagen. Auch der Begriff von Open Source ist Stallman ein Grauen. Er besteht auf seine „Free Software“ um zu zeigen dass seine Software „frei“ ist. Leider wird dieses „frei“ immer mit „umsonst“ verwechselt, deshalb haben andere Leute den Begriff „Open Source“ verwendet, um diese Software zu kategorisieren.

Soweit erst einmal die Betrachtungen über die ursprüngliche Geschichte. Die Erfolgsstory von Open Source beginnt eigentlich erst hier. Im Rahmen der Open Source Bewegung sind so erfolgreiche Projekte wie der Apache Webserver, Sendmail, Mozilla oder aber Open Office entstanden. Wir möchten hier nicht die gesamte Geschichte von Open Source bis zum heutigen Tage ausbreiten, sondern lediglich aufzeigen welche Beweggründe es gab, sich für „Freie Software“ einzusetzen.

3. Unterscheidung der Softwaretypen

Nachdem nun geklärt wurde, woher Open Source kommt, wollen wir jetzt einmal klären, was alles Open Source bzw. Free Software ist.

Software Typ							
Kommerzielle Software							
Probesoftware	<input checked="" type="checkbox"/> (eingeschränkt)	<input checked="" type="checkbox"/>					
Nichtkommerzielle Software	<input checked="" type="checkbox"/> (Nutzungsabhängig)	<input checked="" type="checkbox"/>					
Share ware	<input checked="" type="checkbox"/> (freiwillige Lizenzierung)	<input checked="" type="checkbox"/>					
Royalty-free Binaries („Freeware“)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				
Royalty-free libraries	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Open Source (BSD-Style)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
Open Source (Apache Style)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
Open Source (Linux/GNU Style)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Lizenz Eigenschaften	Keine Anschaffungskosten	Weitergabe erlaubt	Uneingeschränkte Nutzung	Quellcode verfügbar	Quellcode modifizierbar	Öffentliche „Eingaben“ werden angenommen	Alle Veränderungen müssen frei bleiben

Diese Tabelle wurde aus dem so genannten „Halloween Document“ von Microsoft übernommen, dass in dieser Quelle von der „Open Source Initiative“ mit Kommentaren versehen wurde.[3-1].

Erklärungen zu den Lizenz Eigenschaften:

Keine Anschaffungskosten:

Man muss kein Geld bezahlen um die Software benutzen zu können.

Weitergabe erlaubt:

Man darf die Software ohne Auflagen an andere Personen weitergeben.

Uneingeschränkte Nutzung:

Es gibt keine Beschränkungen bei der Nutzung der Software.

Quellcode verfügbar:

Der Quellcode der Software ist einsehbar.

Quellcode modifizierbar:

Jeder darf den Quellcode verändern.

Öffentliche "Eingaben" werden angenommen:

Veränderungen und Vorschläge von dritten Personen werden beim Kernentwicklerteam angenommen.

Alle Veränderungen müssen frei bleiben:

Jede Software, die zumindest Teile eines offenen Programms enthält, muss frei bleiben.

Jetzt werden noch die einzelnen Arten der Software erklärt.

Kommerzielle Software :

Microsofts „Brot und Butter“. Das ist ganz normale Software, die man bezahlen muss und die man normalerweise nur als „Binaries“ bekommt.

Probesoftware:

Software die von Firmen an Interessierte verteilt, um die Software näher zu betrachten. Man bekommt sie umsonst und kann sie weiter verteilen, allerdings hat diese Software viele Einschränkungen und ist nicht voll funktionsfähig.

Nichtkommerzielle Software:

Dies ist Software die zumeist voll funktionsfähig ist und die man kostenlos benutzen kann, solange man sie im privaten Bereich einsetzt. Sobald man sie im geschäftlichen Bereich einsetzt muss man Geld zahlen.

Shareware:

Diese Software ist so ähnlich wie Probesoftware. Allerdings ist sie nicht so stark eingeschränkt. Ziel der Software ist es, dass der Benutzer aus Anerkennung für die Programmierer oder zum frei schalten aller Funktionen eine Lizenz erwirbt.

Royalty-free Binaries („Freeware“):

Diese Freeware wird von dem Programmierer oder Hersteller kostenlos zur Verfügung gestellt. Der Benutzer kann die Software weitergeben und sie uneingeschränkt nutzen. Allerdings hat kein Recht diese zu verändern.

Royalty-free libraries:

Diese freien Bibliotheken sind dazu gedacht, um Leuten zu helfen mit externen Programmen auf ein Programm zugreifen zu lassen. Es wird dadurch erreicht, dass jeder sich die öffentlichen Schnittstellen näher betrachten kann.

Open Source (BSD-Style):

Normalerweise entwickelt ein geschlossenes Team eine Software und gibt sie dann frei, so dass jeder es weiter verändern und verkaufen kann. Allerdings nimmt das Entwicklungsteam im Allgemeinen keine Eingaben und Verbesserungen von anderen Parteien an.

Open Source (Apache-Style)

Genauso wie der BSD-Style, nur dass jetzt auch Verbesserungen von Dritten mit ins Programm einfließen können.

Open Source (Linux/GNU-Style)

Diese Lizenz geht noch einen Schritt weiter. Zusätzlich zu allen bisher genannten Features die auch die anderen Lizenzen haben, kommt hier eine weitere hinzu: man muss jede Software die man aus einem offenen Programm entnimmt, auch wieder frei machen.

4. Diskussionen der Vor- und Nachteile von Open Source

Die Geschichte von Open Source Software (OSS) ist sehr bewegt. Kein anderes Lizenzmodell bzw. keine andere Entwicklungsphilosophie ist so umstritten. Befragt man heutzutage die IT-Welt zu einer Stellungnahme, so läuft man immer wieder Gefahr, einen der berüchtigten flamewars zu entzetteln, also eine erbitterte Diskussion um die Vor- und Nachteile von OSS. Während den Verfechtern meist glühender Fanatismus nachgesagt wird, treten die Gegner oft als Führungspersonal oder Vertreter renommierter Softwareunternehmen auf, die ihr Werk (bzw. das ihrer Programmierer) als geistiges Eigentum betrachten. Somit wird für sie der Quellcode eines Programms zum Grundkapital des Unternehmens und eine Offenlegung käme dem Verrat von Betriebsgeheimnissen an die Konkurrenz und damit dem Bankrott gleich.

Es war und ist nicht einfach für die Open-Source-Gemeinde, die Gegner des Konzepts von den Vorteilen zu überzeugen, die sie darin sehen. Das Prinzip, den Quellcode und damit die Funktionsweise eines Programms für jedermann ersichtlich zu machen, der Programmiersprachen beherrscht, wurde nur allzu oft von beiden Seiten missbraucht, um unterschiedliche Weltanschauungen zu propagieren. So wurde von Seiten der Befürworter von OSS das Konzept gern in Verbindung gebracht mit einer Art „sozialistischer“ IT-Welt, in der niemand ein Monopol auf Informationen haben sollte und Quellcode für jedermann ebenso bei Bedarf erhältlich sein soll wie alle anderen Güter im „wahren Sozialismus“. Dies rief natürlich sofort auch Gegner auf den Plan, die eigentlich mit Software wenig zu tun hatten, aber den Sozialismus nicht guthießen und ihrerseits mit einer „marktwirtschaftlichen“ IT-Welt antworteten. Diese Politisierung des Themas wirkte sich sehr nachteilig auf die Untersuchung von Vor- und Nachteilen von OSS aus und bekräftigte zudem den Kampfgeist der Befürworter.

Es ist daher schwierig, neutrale Beurteilungen zu OSS zu erhalten. Vielfach haben wir auf Studien und Gutachten zurückgegriffen, deren Aussage allerdings von der Meinung der jeweiligen Auftragsgeber beeinflusst wurde.

Verfolgt man die Diskussion um Vor- und Nachteile von OSS, so ergeben sich eine Reihe von Kritikpunkten, die im Laufe der Zeit zusammengetragen wurden. Obwohl sich OSS weiterentwickelt hat und viele der Kritikpunkte inzwischen verschwunden oder geschwächt sind, halten sie sich dennoch hartnäckig als Vorurteile in den Köpfen vieler Entscheidungsträger. Wir haben uns deshalb entschlossen, aus unternehmerischer Sicht die vier wichtigsten Punkte zu hinterfragen, Argumente für beide Positionen zu finden, gegenüberzustellen und ein Fazit zu bilden.

Im Punkt Support unterscheidet sich die Open-Source-Strategie wahrscheinlich am meisten von der proprietären Software, denn der Support läuft hier ganz ähnlich ab wie die Entwicklung, nämlich verteilt. Dies bedeutet, dass mitunter tausende weithin anonyme Programmierer aus der ganzen Welt das Programm weiterentwickeln und warten. Was im privaten Sektor vielleicht zu neuen Freundschaften führt, ist im unternehmerischen Sektor heikel, denn viele Unternehmen sind auf feste Ansprechpartner und teilweise auch auf Vor-Ort-Service und Schulungen angewiesen. Bis vor wenigen Jahren waren weite Teile des OSS-Softwaremarktes diesen Ansprüchen nicht gewachsen.

Allerdings hat sich viel getan. Was früher als Nachteil galt, wird nun als der größte Vorteil von OSS gesehen. Die Entwickler von OSS haben sich inzwischen gut organisiert, wie Projekte wie Mozilla[4-1] oder Apache[4-2] beweisen. Durch die gute Organisation ist es den Projektleitern gelungen, nicht nur die unzähligen Versionen und Modifikationen ihrer Software unter Kontrolle zu bekommen, sondern sich auch hervorragend um Details wie eine ausführliche (technische) Dokumentation, FAQs und Foren zu kümmern, in denen viele Probleme geklärt werden können.

Dass dies bereits erheblich den Aufwand für Einarbeitung und Hilfe senkt, zeigt die Firma ESDC, die ihre Kundendatenbank auf OpenOffice[4-3] umgestellt hat. In ihrer Stellungnahme dazu auf den Seiten von OpenOffice.org[4-4] heißt es: „[...] durch die direkte Integration der Datenbank in ein Office-Paket können auch durchschnittliche AnwenderInnen mit sehr wenig Schulungsaufwand auf alle Features erfolgreich zugreifen.“

Auch scheint der Markt für Support selbst recht groß zu sein, wie der Marktanalyst Soreon zeigt. In dessen Studie zum Open-Source-Markt in Deutschland[4-5] heißt es in einem Resümee: „Die größte Signifikanz haben Umsätze aus Support-Services gefolgt von Software-Training. „

Inzwischen sind deshalb auch die großen Unternehmen nach und nach an OSS interessiert und nehmen sie in ihr Repertoire auf. Nach IBM hat nun auch Novell den Einstieg mit Linux geschafft[4-6], und auch SUN verkauft inzwischen Linux für seine Workstation und Server[4-7]. Ebenfalls sind viele Produkte, die vorher als proprietäre Software verkauft wurden, nun unter neuer Flagge als Open Source zu haben. Nachdem Apple den Quellcode für Darwin 7.0, die grafische Oberfläche von MacOS 10.3, freigegeben hat[4-8], entschied sich auch Oracle, den 9i Server (zumindest unter Linux) als Open Source laufen zu lassen. Der IT-Nachrichtendienst Golem schreibt dazu: „Oracle bietet darüber hinaus weltweit Support für seine

Produkte unter Linux ohne zusätzliche Kosten an. Dazu beschäftige Oracle allein ein Team von 6.000 Mitarbeitern. Mit dem geplanten Oracle Open Source Development Center plant Oracle zudem einen Online-Dienst über den Entwickler mit Software, Beispiel-Code und Anleitungen versorgt werden sollen. “[4-9].

Damit geht der Trend klar in Richtung OSS bzw. der Ausnutzung der Vorteile, die verteiltes und offenes entwickeln von Software auf den Support haben.

4.b Vor- und Nachteile von Open Source

Kosten

Der Punkt, der wahrscheinlich am ehesten OSS zugesprochen wird, ist der Kostenvorteil. Zumindest im Anschaffungspreis steht hier OSS mit wenig oder gar keinen Kosten gegen die restriktive proprietäre Software, für die meist nicht nur die Anschaffungs-, sondern auch Lizenzkosten stark ins Gewicht fallen.

Aber diese Faktoren sind nur der Tropfen auf den heißen Stein: Den größten Faktor zur Berechnung des Einsparungspotentials von Software bilden Schulung und Einarbeitungszeit. So hat die Firma Microsoft beim Marktanalysten IDC eine Studie zum Kostenvorteil von Microsoft Betriebssystemen gegenüber Open-Source-Betriebssystemen (speziell Linux) in Auftrag gegeben[4-10]. Deren Ergebnis war nicht überraschend: Zwar sind die Anschaffungs- und Lizenzkosten bei Microsoft höher, jedoch rechnet sich das mit dem Aufwand für Schulung und Support, der laut IDC bei Open-Source-Lösungen zeitlich wie personell höher liege, somit sei Microsoft über einen Zeitraum von fünf Jahren gesehen die günstigere Alternative.

Demgegenüber steht die Studie „Kassensturz“[4-11] des Marktanalysten Soreon. Das Resümee von Soreon sagt aus, dass durch den Einsatz von OSS Einsparungen von bis zu 30% möglich sind. Allerdings wirkt sich das auf große Unternehmen stärker aus als auf kleinere, weil die größeren meist über eigenes Open-Source-Know-How verfügen und hier die wegfallenden Lizenzkosten stärker ins Gewicht fallen. Dennoch sei eine gründliche Überprüfung für jedes Unternehmen vor der Umstellung angebracht, eine generelle Empfehlung wollte Soreon nicht geben.

Auch die Landeshauptstadt München hat vor ihrer Umstellung auf Open Source eine Studie[4-12] diesbezüglich beim Unternehmensberater Unilog in Auftrag gegeben, die, was die Kosten angeht, zunächst auf ähnliche Ergebnisse kommt wie die IDC-Studie von Microsoft, nämlich dass der Einsatz von OSS zunächst (geringfügig) teurer ist als das Beibehalten bzw. Erneuern der vorhandenen proprietären Software. Allerdings, und das steht nur in einer Fußnote, erwähnen die Verfasser, dass die Sicht nur auf die folgenden fünf Jahre beschränkt sei und dass eine Betrachtung über einen Zeitraum von zehn Jahren oder mehr die langfristigen Kostenvorteile von OSS aufzeigen würde. Genau dies gilt es nämlich zu bedenken, wenn man über den Einsatz von OSS innerhalb eines Unternehmens diskutiert. Alle hier genannten Studien haben einen Betrachtungszeitraum von maximal fünf Jahren und sind damit unzureichend, wenn es um die langfristigen Vorteile von OSS geht.

Der oft erwähnte hohe Schulungsaufwand ist nämlich nur am Anfang gegeben, jedoch langfristig gesehen ist OSS die bessere Alternative, da z.B. die Bedienoberfläche nicht unbedingt nach kommerziellen (und damit z.T. unsinnigen) Maßstäben gefertigt wird, sondern nach funktionalen und psychologischen Aspekten, was beispielsweise die Wiederauffindung von Funktionen eines Programms bzw. die Orientierung extrem erleichtern kann (gutes Beispiel ist das Herunterfahren von Windows, wofür man zunächst Start drücken muss).

Somit ist festzuhalten, dass für eine ausführliche Kostenanalyse des Einsatzes von OSS in Unternehmen eine kurzfristige Betrachtung nicht ausreicht, da die Einsparpotentiale erst langfristig auftreten.

4.c Vor- und Nachteile von Open Source

Abhängigkeit

Die Angst, von einem oder wenigen Softwareherstellern abhängig zu sein, ist für viele fanatische Open-Source-Verfechter eines der Hauptargumente für einen Wechsel zu OSS. Schon länger kursieren Gerüchte, dass der Marktführer Microsoft seine Monopolstellung ausnutze, um vertrauliche Daten zu sammeln. Besonders das Betriebssystem Windows XP ist seit der Markteinführung mit großer Skepsis untersucht worden, weil es in Verdacht stand, allzu oft „nach hause zu telefonieren“, also insgeheim Verbindung mit den Microsoft-Servern aufzunehmen. Die meisten dieser Gerüchte konnten zwar entkräftet werden, doch die Skepsis blieb. Auch die noch nicht erschienene Nachfolgerversion mit dem Codenamen „Longhorn“ sorgt mit seiner Unterstützung für das ebenfalls heiß diskutierte Palladium-Modul für Aufruhr unter den Datenschützern.

Allerdings hat die „Alles-aus-einer-Hand“-Philosophie des Monopolisten auch Vorteile, werden doch dadurch auch Patches und Bugfixes vom Hersteller selbst geliefert und das „problemlose“ Zusammenarbeiten von verschiedenen Programmen desselben Herstellers ersparen Ärger und Troubleshooting. So schreibt der Heise Verlag dazu in seinem Newsticker: „Die Frankfurter Oberbürgermeisterin Petra Roth unterzeichnet morgen zusammen mit Jürgen Gallmann, Chef von Microsoft Deutschland, einen Rahmenvertrag über die Software-Ausstattung der Verwaltung in der Mainmetropole. Laut Thomas Scheben, Leiter der dortigen Presseabteilung, geht es unter anderem darum, die diversen Computer der Dezernate und Ämter auf den aktuellen Stand zu bringen. Bislang liefen auf den diversen PCs bis zu vier verschiedene Generationen des Microsoft-Betriebssystems. So soll die gesamte Verwaltung der Stadt Frankfurt bei fälligen Updates und Patches sofort auf den neuesten Stand gebracht werden.“[4-13].

Kritisch ist die Herstellerabhängigkeit allerdings doch, da die Schnittstellen meist ebenfalls proprietär sind und somit die Produkte einer Firma meist nur mit den Produkten derselben Firma kooperieren. Diese Schnittstellenproblematik wurde vom Marktforschungsunternehmen TNS Emnid aufgegriffen und führte zu einer Zusammenfassung aller diesbezüglichen Forenbeiträge, die über 500 IT-Spezialisten aus aller Welt dort zusammengetragen hatten[4-14]. Der Nachrichtendienst Golem schreibt darüber: „Die Voraussetzungen für Neuinvestitionen sehen IT-Profis dann am ehesten gewährleistet, wenn vorhandene Standards weiter verbessert und offenere Systeme weiterentwickelt bzw. Schnittstellenprobleme vermindert werden. [...] Hier zeigt sich eines der Hauptprobleme der IT-

Industrie. Um an möglichst vielen Stellen der Wertschöpfungskette zu partizipieren, sind die Systeme nicht ausreichend kompatibel. Das mag zum einen an der teilweise hohen Komplexität der jeweiligen IT-Lösung liegen, zum anderen werden aber auch mehr oder weniger bewusste Abhängigkeiten geschaffen, um auch das Folgegeschäft mit seinen Kunden zu sichern', so Wolfgang Best, Director Telecommunication & IT bei TNS Emnid.“ Daraus ist ersichtlich, dass die Offenlegung der Schnittstellen eines Programms dessen Nutzbarkeit erheblich erhöhen kann.

Auch die Landeshauptstadt München hat ihre Entscheidung für den Umstieg zu OSS letztlich aus Gründen der Herstellerunabhängigkeit getroffen. So meint Oberbürgermeister Ude: "Mit diesem richtungsweisenden Grundsatzbeschluss sichert sich München nicht nur als erste deutsche Großstadt eine größere Herstellerunabhängigkeit ihrer IT-Infrastruktur, sondern setzt auch ein klares Zeichen für mehr Wettbewerb im Software-Markt." [4-15]. Aus der oben erwähnten Studie der Landeshauptstadt [4-12] geht außerdem klar hervor, dass die IT-Abteilung der Stadt durch den Einsatz von Microsoft-Produkten (speziell Windows) „[...] einen hohen Migrationsdruck in Richtung auf weitere MICROSOFT Produkte [...]“ erfahre, sodass „schon die Clients unter dem Betriebssystem WindowsXP [...] fast nicht mehr sinnvoll und sicher ohne entsprechende Microsoft Backoffice-Dienste betrieben werden“ können.

Somit wird deutlich, dass der Verlust der Herstellerabhängigkeit einen höheren Wettbewerb für Softwarehersteller und damit meist Kostenersparnis für Unternehmen bedeutet.

Dieser in der privaten IT-Welt eher untergeordnete Punkt spielt sowohl für Unternehmer wie auch Softwarefirmen eine umso größere Rolle. Allzu oft wurde der Vorwurf deutlich, die GPL würde die Rechte der Urheber nur ungenügend schützen und sei als Lizenz ungeeignet, da der Urheber dann keine Kontrolle mehr über die Verbreitung seiner Software habe.

Diese Kritik zu widerlegen stellt sich schwieriger dar als bei den anderen Punkten. So existieren weiterhin erhebliche Probleme beim Umgang mit der GPL, da sie eine Lizenz ist, die ursprünglich nicht zur kommerziellen Nutzung von Software vorgesehen war. Die Bedingung, dass GPL-lizenzierte Software (also solche ohne Lizenzgebühren) auch weiterhin unter der GPL veröffentlicht werden muss, bringt viele findige Informatiker in die Zwickmühle: einerseits haben sie eine gute Idee zur Weiterentwicklung offener Software, andererseits verbietet Ihnen die GPL jegliche kommerzielle Nutzung in Form von Lizenzen. Ebenfalls problematisch ist die Mischung von proprietärer und offener Software, z.B. auf einem Datenträger. Welche Lizenzform dann überwiegt und ob überhaupt ein Kompromiss gefunden werden kann (darf), ist rechtlich noch völlig offen.

Die Meinungen zur Rechtssicherheit der GPL und anderer Open-Source-Lizenzen variieren je nach Lager und Interessensgruppe. Eine vom Verband der Softwareindustrie Deutschlands (VSI) bei Prof.Dr. Gerhard Spindler in Auftrag gegebene Studie „Rechtsfragen der Open Source Software“[4-16] deckt erhebliche Mängel der GPL im Hinblick auf die Vermarktung von OSS auf. Dazu der Nachrichtendienst Golem: „Die Bestimmungen der GPL weisen laut der VSI-Studie erhebliche Rechtsunsicherheiten auf, die nach Ansicht des VSI im Wesentlichen auf die Grundprinzipien von freier Software zurückzuführen sind, insbesondere die freiwillige Weitergabe der Programmierleistungen, die nicht zentral unter der Leitung eines Unternehmens entwickelt worden sind.“ Weiter noch geht die WIPO, die World Intellectual Property Organization. Sie ist der Meinung, die GPL stehe im Gegensatz zu den Zielen der WIPO, geistiges Eigentum zu schützen[4-17].

Zu dieser Studie hat das ifrOSS (Institut für Rechtsfragen der Freien und Open Source Software) ausführlichst Stellung bezogen[4-18]. Sie sind der Ansicht, die GPL funktioniere in ihrer Funktion als Softwarelizenz, wie §1 bereits zeigt „Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, dass Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluss veröffentlichen, alle Vermerke, die sich auf diese

Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und des Weiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.“[4-19]. Außerdem, so ifrOSS, greifen auch Hersteller proprietärer Software auf Open Source zurück und begäben sich damit ebenfalls in die vermeintliche Rechtsunsicherheit.

Viele Firmen und Verwaltungen wechseln hauptsächlich wegen der wegfallenden Lizenzkosten zu OSS. So schreibt zumindest die OVIDIUS Berlin GmbH nach ihrem Umstieg auf OpenOffice in deren Storyboard: „Die Gründe [für den Umstieg, Anm.d.Verf.] waren nicht ausschließlich finanzieller Art - wir haben kein Problem damit, für die von uns genutzte Software zu bezahlen. Wichtiger war der zu hohe Verwaltungsaufwand, der benötigt wird, um mit den Lizenzen immer rechtlich auf der sicheren Seite zu sein.“[4-4].

Weiterhin schrieb die Schule Wengen: „Es ist uns nun möglich, den Schülern eine CD mit dem hier benutzten Office-Programm und diversen Dokumenten gratis mit nach Hause zu geben. So können Hausaufgaben mit dem Office-Programm, das in der Schule in Gebrauch ist, gemacht werden. Da sich OpenOffice.org bequem neben MS-Word installieren lässt, beharren die Schüler in den wenigsten Fällen auf MS-Word, was aber dank OpenOffice.org auch kein Problem ist. [...] Ohne Zweifel ermöglicht uns erst OpenOffice.org und Linux, die frei (ohne irgendwelche Rechte zu verletzen) kopiert werden dürfen, trotz unserer beschränkten Mitteln eine hochwertige Informatikinfrastruktur aufzubauen.“[4-4]

Abgesehen von allen Argumenten für oder gegen diese Lizenz ist die GPL (zumindest bei Neukreationen) eine freiwillige Lizenz, wer also sein Programm unter der GPL veröffentlicht, weiß in der Regel, welche Konsequenzen das hat und ist mit ihnen einverstanden. Somit erübrigt sich die Frage, ob die GPL den Vorstellungen des Urhebers gerecht wird oder nicht.

5. Zusammenfassung

Wenn man die ganzen Punkte näher betrachtet, fällt auf, dass es immer noch eine Menge Vorurteile gegen Open Source gibt. Diese sind - berechtigt oder nicht - immer noch ein Grund, warum Open Source noch Probleme hat, sich weiter durchzusetzen. Zum einen liegt das mit Sicherheit daran, dass Open Source gerade in der Anfangszeit viele technische Probleme hatte und teilweise heute noch hat. Ein anderer Grund dürfte allerdings auch sein, dass es einfach nicht so eine große Lobby für Open Source gibt wie für proprietäre Software.

Weiterhin gibt es noch rechtliche Probleme. So kann man nur sehr schwer jemanden finden, den man für die Fehlfunktionen der Software haftbar machen könnte. Außerdem zeigt der absurde Gerichtsstreit zwischen IBM und SCO dass (noch) nicht alle rechtlichen Fragen geklärt sind. Hinzu kommt noch die Problematik Softwarepatente. Es ist eher unwahrscheinlich, dass unabhängige Programmierer die Übersicht darüber haben, welche Aspekte der Software nun rechtlich geschützt sind und welche nicht.

Es gibt außerdem einige Bereiche, wo es wenig sinnvoll ist, Open Source einzusetzen. Diese Bereiche sind zum Teil sehr komplex und erfordern ein profundes Fachwissen. Da werden sich dann kaum Leute finden, die bereit sind, sich in die Thematik einzuarbeiten. Solche Bereiche könnten zum Beispiel CAD-Software oder spezielle medizinische Software sein.

Ein weiterer nicht zu unterschätzender Aspekt ist die GPL. Diese Grundlage für freie Software wurde von Richard Stallman geschrieben, um Software auf jeden Fall frei zu halten. So muss jede selbst erstellte Version einer Software auch frei bleiben und auch jedes Fragment von freier Software, das man in sein eigenes Programm übernimmt, würde das eigene Programm zu freier Software machen. Dies kann definitiv nicht im Sinne vieler Firmen sein, die Open Source Software modifizieren und einsetzen wollen. Hier müssen alternative Lizenzen geschaffen werden, bei denen Firmen (gegen entsprechende Bezahlung) mehr Rechte über die Software erlangen können.

Trotz all der oben genannten Probleme hat Open Source aber auch etliche positive Seiten. Eine der wichtigsten ist der enorme Innovationsgrad. Die vielen Menschen, die freiwillig mitarbeiten, machen das aus eigenem Antrieb und haben folglich eine höhere Motivation als Mitarbeiter, die für eine Firma programmieren müssen. Außerdem haben viele Projekte mehr Mitarbeiter als sich eine Firma leisten könnte. Dadurch gibt es natürlich mehr Sichtweisen auf das Programm, und jeder kann die Fortschritte von anderen begutachten und eigene Vorschläge einbringen.

Ein weiterer wichtiger Punkt ist die Unabhängigkeit der Open Source Projekte. Da sie nicht in erster Linie gewinnorientiert arbeiten müssen, können sie sich die nötige Zeit nehmen, um die Software fertig zu entwickeln. Die Software ist auch nicht von einer Firma und ihrem Wohlergehen abhängig, sondern kann auch von beliebigen anderen Programmierern weiterentwickelt werden. Außerdem kann man Projekte, die nicht auf Gewinn aus sind, nicht mit den üblichen Mitteln der großen Konzerne aus den Markt drängen.

Des Weiteren muss man auch anerkennen, dass sich Open Source Software langsam in Sachen Qualität und Ausgereiftheit an Proprietäre Software angeglichen hat bzw. diese sogar überholt hat. Dies ist natürlich für Microsoft - den absoluten Monopolisten im Softwaremarkt - die unangenehme Erkenntnis, dass kostengünstigere Software, die noch dazu den Microsoftprodukten mindestens ebenbürtig sind, immer mehr Marktanteile erobert.

Trotz der vielen Widersprüche, die im Laufe des Vortrages in Bezug auf Open Source aufgetreten sind, lässt sich zumindest folgendes feststellen: Open Source ist in vielen Fällen zu einer vollwertigen Alternative zu den bisherigen Produkten geworden und wird auch bei vielen Firmen ernsthaft in Erwägung gezogen. Trotzdem gibt es für Open Source noch viele Hürden, die verhindern, dass Open Source in sämtlichen Bereichen ein gleichwertiger Gegner zu proprietärer Software wird. Ein wichtiger Schritt wäre sicherlich die vollständige Klärung der dringendsten Rechtsfragen, und es ist zu hoffen, dass sich Open Source Software auch in den Privathaushalten immer mehr durchsetzt. Weiterhin müssen vor allem in Bezug auf die Lizenzen Alternativen geschaffen werden, da die GPL für eine sinnvolle wirtschaftliche Nutzung der Software nur bedingt zu gebrauchen ist. Hier müsste man Alternativen finden, wie z.B. „Paid Source“ bzw. „Pay-for-Source“, wo nur die Firma den Sourcecode erhält, die ihn bezahlt hat, und diesen Sourcecode dann für Zwecke zu verändert und ihn dann aber nicht mehr freigeben muss.

Außerdem muss man abwarten, inwieweit man mit Open Source Geld verdienen kann. Denn nur wenn man damit viel Geld verdienen kann, sind da auch große Firmen, die Geld in Open Source stecken. Und es ist fraglich, ob Open Source mehr ist, als ein (recht erfolgreicher) Versuch, Microsoft Marktanteile abzunehmen.

Quellenverzeichnis

Kapitel 2: Geschichte von „Open Source“

- [2-1] Freyermuth, Gandolf S.: „Offene Geheimnisse“.c't-Online:
<http://www.heise.de/ct/01/20/176/> (03.01.2004)
- [2-2] Dinkelman, Carsten; Meyer:, Marko „Überblick über das Betriebssystem Linux“.FH Zwickau:
http://wwwstud.fh-zwickau.de/~linux/projekte/Linux-Kurs/Linux-kurs_html/node6.html (03.01.2004)
- [2-3] Weber, Helmut: „Zur Geschichte von UNIX“.FH Wiesbaden:
<http://wwwsys.informatik.fh-wiesbaden.de/sysprog/buch0017.htm> (03.01.2004)
- [2-4] „The Creation of the UNIX Operating System“.Lucent Technologies:
<http://www.bell-labs.com/history/unix/chaos.html> (03.01.2004)
- [2-5] Witt, Frank, H.: „Globalisierung und Regulierung des Medien- und Telekommunikationsektors“.Universität Paderborn:
http://iug.uni-paderborn.de/0x83ea6065_0x000094ec (03.01.2004)
- [2-6] Möller, Erik: „Die Reformation zum Anfassen: GNU/Linux und Open Source“.Telepolis:
<http://www.heise.de/tp/deutsch/inhalt/te/9786/1.html> (03.01.2004)
- [2-7] „Berkeley Software Distribution“.Wikipedia.de:
http://de.wikipedia.org/wiki/Berkeley_Software_Distribution (03.01.2004)
- [2-8] „Open Source kurz & gut“.O'Reilly
http://www.oreilly.de/german/freebooks/os_tb/os_tb_1.htm (03.01.2004)
- [2-9] Stallman, Richard: „Ursprüngliche Ankündigung“.www.gnu.org:
<http://www.gnu.org/gnu/initial-announcement.de.html> (03.01.2004)
- [2-10] Stallman, Richard: „Das GNU Projekt“. www.gnu.org:
<http://www.gnu.org/gnu/thegnuproject.de.html> (03.01.2004)
- [2-11] Hasan, Ragib:„History of Linux“
<http://ragib.hypermart.net/linux> (03.01.2004)
- [2-12] Stallman, Richard: „Linux und das GNU-Projekt“. www.gnu.org:
<http://www.gnu.org/gnu/linux-and-gnu.de.html> (03.01.2004)

Kapitel 3: Unterscheidung der Softwaretypen

- [3-1] „Halloween Document I“.Open Source Initiative:
<http://www.opensource.org/halloween/halloween1.html> (03.01.2004)

Kapitel 4: Vor- und Nachteile von Open Source

- [4-1] Das Mozilla-Projekt. www.mozilla.org (03.01.04).

- [4-2] Das Apache-Projekt. *www.apache.org* (03.01.04).
- [4-3] Das OpenOffice-Projekt. *www.openoffice.org* (03.01.04).
- [4-4] „Erfolgsgeschichten und Anwenderberichte“.OpenOffice.org:
http://de.openoffice.org/marketing/stories/index.html (03.01.2004).
- [4-5] Studie „The Market for Open Source Software in Germany“.Soreon Research GmbH:*http://www.soreon.de/html/studien/opensource_ergebnisse.htm* (03.01.04).
- [4-6] Egle, Christian:„Novell unterzeichnet Vereinbarung zur Übernahme von SUSE LINUX“.SuSe Linux AG:
http://www.suse.de/de/company/press/press_releases/archive03/novell_suse.html (03.01.04).
- [4-7] Ihlenfeld, Jens:„Sun: Linux ist reif für den Desktop“.Golem.de:
http://www.golem.de/0308/26817.html (03.01.04).
- [4-8] Ihlenfeld, Jens:„Apple veröffentlicht Source Code von Darwin 7.0“.Golem.de:
http://www.golem.de/0310/28202.html (03.01.04).
- [4-9] Ihlenfeld, Jens:„Oracle setzt voll auf Linux“.Golem.de:
http://www.golem.de/0308/26859.html (03.01.04).
- [4-10] Jean Bozman, Al Gillen, Charles Kolodgy, Dan Kusnetzky, Randy Perry, David Shiang:„Windows 2000 vs. Linux für Unternehmensanwendungen“.IDC: Framingham,MA 2002.
- [4-11] Studie „Kassensturz“.Soreon Research GmbH:
http://www.soreon.de/html/studien/kassensturz_ergebnisse.htm (03.01.04).
- [4-12] „Kurzfassung der Clientstudie der LHM 2002“.Unilog Integrata:München 2002.
- [4-13] Wilkens,Andreas:„Frankfurt setzt weiter auf Microsoft“.Heise Newsticker:
http://www.heise.de/newsticker/data/anw-27.05.03-001/ (03.01.04).
- [4-14] Donath,Andreas:„Software-Standards und -Öffnung sind IT-investitionsfördernd“. Golem.de:*http://www.golem.de/0310/27776.html* (03.01.04).
- [4-15] Ude, Christian: Kommentar zur IT-Umstrukturierung der LHM:
http://www.muenchen.de/aktuell/ms_linux.htm (03.01.04).
- [4-16] Ihlenfeld, Jens:„VSI: Einsatz von Open Source rechtlich kritisch“.Golem.de:
http://www.golem.de/0307/26226.html (03.01.04).
- [4-17] Ihlenfeld, Jens:„Microsoft stoppt internationalen Open-Source-Gipfel“.Golem.de:
http://www.golem.de/0308/27075.html (03.01.04).
- [4-18] Ihlenfeld, Jens:„ifrOSS: VSI zieht voreilige Schlüsse bezüglich Open Source“. Golem.de:*http://www.golem.de/0307/26288.html* (03.01.04).
- [4-19] Katja Lachmann,Peter Gerwinski:„GNU General Public License, Deutsche Übersetzung der Version 2“. G-N-U GmbH:Essen 2000.