

**Open Source –  
Prozesse,  
ökonomische Implikationen  
und Generalisierung**

3-Monats-Arbeit im Rahmen der Prüfung  
für Diplom-Wirtschaftsinformatiker  
an der Universität Göttingen

vorgelegt am 1. November 2001  
von Frank Bauer  
aus Witzenhausen

# Vorwort

Obwohl ein Vorwort für eine Diplomarbeit unüblich sein mag, möchte ich dennoch die Gelegenheit nutzen, vorab einige persönliche Dinge anzumerken.

Die Idee für dieses – vermutlich recht ungewöhnliche – Diplomarbeitsthema hatte ich vor ungefähr eineinhalb Jahren. Bedingt durch die zunehmende Unzufriedenheit über Microsoft Windows begann ich damals, mich nach Alternativen umzusehen. Bald darauf schon hatte ich erstmals GNU/Linux auf meinem Rechner installiert.

Etwa ein halbes Jahr später begann ich damit, einen Internet- und File-Server für ein kleines privates Netzwerk einzurichten. Als Betriebssystem für diesen Rechner wählte ich eben jenes GNU/Linux. Die für mich als Einsteiger problemlose Einrichtung verschiedener Netzwerkdienste wie IP-Masquerading und File-Sharing sowie die Tatsache, dass dieser Rechner nach nunmehr einem knappen dreiviertel Jahr täglich beinahe rund um die Uhr noch immer absolut einwandfrei seinen Dienst verrichtet, überzeugten mich endgültig von diesem Betriebssystem und den dazugehörenden Applikationen.

Mit wachsender Begeisterung stellte sich mir nun irgendwann die Frage, wie es möglich ist, dass eine derart – wie ich meine – qualitativ hochwertige Software, deren Entwicklung für Unternehmen mit immensen Kosten verbunden wäre, von vielen, weltweit verteilten Menschen *einfach so* geschrieben und völlig kostenlos aus dem Internet bezogen werden kann<sup>1</sup>.

So entschloss ich mich dazu, mehr über freie bzw. Open Source Software in Erfahrung zu bringen. Der Enthusiasmus, mit dem mehr und mehr Menschen in ihrer Freizeit scheinbar völlig uneigennützig derartige Software schreiben und veröffentlichen, obwohl sie mitunter mit derselben Tätigkeit in (Software-)Unternehmen traumhafte Gehälter erzielen könnten, faszinierte mich mit der Zeit zunehmend. Aber damit nicht genug: Diese Menschen ermuntern auch noch ausdrücklich dazu, ihre Software weiterzugeben, deren Quellcode den eigenen Bedürfnissen anzupassen und somit Verände-

---

<sup>1</sup> Volker Grassmuck, der einige sehr lesenswerte Beiträge zu diesem Thema veröffentlicht hat, schreibt in diesem Zusammenhang: “Dass Menschen in freier Kooperation ohne primäres Interesse am Gelderwerb, ohne sich jemals getroffen zu haben und ohne eine Leitung, die ihnen sagt, was sie zu tun haben, hochwertige Software schreiben, ist eine erstaunliche Tatsache.” Grassmuck (2001a). Dem ist nichts mehr hinzuzufügen.

rungen an ihr vorzunehmen – eine für Windows & Co. undenkbare Vorstellung. Je mehr ich über die hinter dieser Art Software stehende Überzeugung sowie ihren revolutionären Charakter las und je näher der Zeitpunkt der Themenvergabe für die anstehende Diplomarbeit kam, desto stärker stand für mich mein Thema fest.

Nach dieser Entscheidung galt es natürlich zu klären, ob sich jemand an der Georg-August-Universität Göttingen findet, der sich als Betreuer hierfür bereit erklären würde. Gefunden habe ich diesen Betreuer in Joachim Rawolle, wissenschaftlicher Mitarbeiter am Lehrstuhl für Wirtschaftsinformatik, Abteilung II von Professor Schumann, dem ich dafür an dieser Stelle danken möchte.

Auch ich kann es mir also nicht nehmen lassen, einigen Personen meinen Dank auszusprechen. Zunächst sind hier meine Eltern zu nennen, die mir während der letzten drei Monate jeden erdenklichen Freiraum ließen. Zudem möchte ich mich bei meinen Freunden und Mitbewohnern Jörg Schmidt und Christoph Swonke bedanken, die mir meine mitunter nicht zum Besten bestellte Laune während dieser Zeit niemals verübelten. Dasselbe gilt selbstverständlich für die übrigen Jungs und Mädels aus Göttingen, Kassel, Melsungen und Hessisch Lichtenau, bei denen ich mich vereinzelt sträflich lange nicht gemeldet habe. Besonders erwähnen möchte ich noch Mario Kolb, denn von ihm habe ich wertvolle Hinweise erhalten, über die ich sonst möglicherweise gestolpert wäre. Und last, but by no means least möchte ich mich bei Susanne bedanken – du weißt, wofür.

Mit der Zeit wurde zunehmend offensichtlicher, dass sich die Frage nach *frei oder nicht-frei* für viele Open Source-Anhänger nicht nur auf Software beschränkt. Obwohl es nicht immer einfach war, sich einer ideologisch geprägten Argumentation und Dokumentation zu entziehen, habe ich mich stets um Objektivität bemüht. Ob mir dies jederzeit gelungen ist, wird sich herausstellen müssen.

Wenn in dieser Arbeit von dem Anwender, dem Entwickler, dem Teilnehmer etc. die Rede ist, so ist damit selbstverständlich auch der weibliche Teil der Bevölkerung gemeint. Ständig *Anwenderinnen und Anwender* bzw. *AnwenderInnen* o.ä. vor Augen zu haben, trägt jedoch nicht gerade zur Lesbarkeit bei, weshalb ich mich in meinen Ausführungen auf den maskulinen Genus bei der Bezeichnung von Personengruppen und -typen beschränke.

Obwohl es nicht gerade zu meinen Stärken zählt und ich ungefähr ein halbes Jahr vor Beginn dieser Arbeit erstmals ein Dokument mit L<sup>A</sup>T<sub>E</sub>X erstellte, hat es dieses Thema fast von selbst verboten, auf Microsoft Word oder dergleichen zurückzugreifen. Auch wenn es auf den ersten Blick kompliziert aussehen und die Vielzahl der Befehle abschrecken mag, kann ich dennoch jedem nur empfehlen, L<sup>A</sup>T<sub>E</sub>X einer genaueren

Untersuchung zu unterziehen – ich jedenfalls habe nur positive Erfahrungen damit gemacht.

Dem Gedanken freier Software folgend wird diese Arbeit unter <http://www.opentheory.org> veröffentlicht.

COPYRIGHT © 2001 FRANK BAUER.

Es wird die Erlaubnis gewährt, dieses Dokument zu kopieren, zu verteilen und/oder zu verändern unter den Bedingungen der GNU Free Documentation License, Version 1.1; mit dem unveränderlichen Abschnitt *Vorwort* sowie keinen Vorder- bzw. Rückseitentexten. Eine Kopie dieser Lizenz ist in dem Abschnitt enthalten, der mit *Anhang A: GNU Free Documentation License* betitelt ist.

*Göttingen, im Oktober 2001*

*Frank Bauer*

*If I am not for myself, who will be for me?*

*If I am only for myself, what am I?*

*If not now, when?*

– *Richard M. Stallman*

*Dedicated to Felix Back. May the force be with you.*

# Inhaltsverzeichnis

<b>Abkürzungsverzeichnis</b>	<b>VI</b>
<b>1 Einleitung</b>	<b>1</b>
<b>2 Grundlagen</b>	<b>3</b>
2.1 Open Source – Definition, Begriffe, Abgrenzung und Lizenzen . . . . .	3
2.1.1 The Open Source Definition . . . . .	3
2.1.2 Ergänzende Begriffsbestimmungen . . . . .	6
2.1.3 Abgrenzung freier Software von nicht-freier Software . . . . .	7
2.1.4 Softwarelizenzen . . . . .	9
2.2 Historische Entwicklung freier Software und der Open Source Software- Bewegung . . . . .	11
2.3 Ausgewählte Open Source Software-Projekte . . . . .	15
2.3.1 Apache . . . . .	15
2.3.2 Sendmail . . . . .	16
2.3.3 Perl . . . . .	16
<b>3 Open Source Software-Projekte</b>	<b>18</b>
3.1 Übergreifende Aspekte . . . . .	18
3.1.1 Spezifische Eigenschaften . . . . .	18
3.1.2 Motivationen von Teilnehmern . . . . .	19
3.1.3 Voraussetzungen . . . . .	22
3.1.4 Qualitätsmanagement . . . . .	24
3.1.5 Konfliktmanagement . . . . .	28
3.2 Organisation . . . . .	29
3.2.1 Aufbauorganisation und Konfiguration . . . . .	30
3.2.2 Ablauforganisation und Koordination . . . . .	31
3.3 DV-technische Unterstützung . . . . .	36
3.3.1 Kommunikationsmittel . . . . .	36
3.3.2 Concurrent Versions System . . . . .	37
3.3.3 SourceForge.net . . . . .	39

3.4	Modell des Verlaufs eines Open Source Software-Projekts . . . . .	40
3.4.1	Ereignisgesteuerte Prozessketten . . . . .	41
3.4.2	Modell eines Projektverlaufs . . . . .	42
<b>4</b>	<b>Ökonomische Implikationen</b>	<b>45</b>
4.1	Auswirkungen auf bestehende Geschäftsmodelle . . . . .	46
4.1.1	Vorteile und Chancen durch Open Source Software . . . . .	46
4.1.2	Nachteile und Risiken durch Open Source Software . . . . .	51
4.2	Open Source Software als Gegenstand der Geschäftstätigkeit . . . . .	53
4.3	Möglichkeiten der Beteiligung an Open Source Software-Projekten . .	57
4.4	Übertragbarkeit organisatorischer Aspekte des Open Source-Modells auf Unternehmen . . . . .	58
4.4.1	Unternehmensübergreifende Netzwerke . . . . .	58
4.4.2	Unternehmensinterne Arbeitsorganisation . . . . .	59
<b>5</b>	<b>Generalisierung</b>	<b>61</b>
5.1	Technische Übertragbarkeit . . . . .	62
5.2	Organisatorische Übertragbarkeit . . . . .	63
5.3	Rechtliche Übertragbarkeit . . . . .	63
5.3.1	Exkurs: Konformität der GPL mit dem deutschen Urheber- rechtsgesetz . . . . .	63
5.3.2	Open Content-Lizenzen . . . . .	64
5.4	Fazit . . . . .	65
<b>6</b>	<b>Fazit und Ausblick</b>	<b>67</b>
	<b>Literatur- und Quellenverzeichnis</b>	<b>VIII</b>
	<b>Anhang A: GNU Free Documentation License</b>	<b>XVIII</b>
	<b>Erklärung</b>	<b>XXVI</b>

# Abkürzungsverzeichnis

<b>Abkürzung</b>	<b>Bedeutung</b>
Abb.	Abbildung
AFPL	Aladdin Free Public License
aktual.	aktualisiert
ARPA	Advanced Research Projects Agency
Aufl.	Auflage
Bd.	Band
BMWi	Bundesministerium für Wirtschaft und Technologie
bspw.	beispielsweise
bzgl.	bezüglich
bzw.	beziehungsweise
ca.	circa
CPAN	Comprehensive Perl Archive Network
CVS	Concurrent Versions System
d.h.	das heißt
durchges.	durchgesehen
DV	Datenverarbeitung
eMail	electronic Mail
evtl.	eventuell
etc.	et cetera
FSF	Free Software Foundation
ggf.	gegebenenfalls
GNU	GNU's Not UNIX
GPL	GNU General Public License
Hrsg.	Herausgeber
i.d.R.	in der Regel
IETF	Internet Engineering Task Force
insb.	insbesondere
IPL	InterBase Public License

---

<b>Abkürzung</b>	<b>Bedeutung</b>
KBSt	Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik
KDE	K Desktop Environment
LGPL	GNU Lesser General Public License
Mio.	Millionen
MIT	Massachusetts Institute of Technology
MPL	Mozilla Public License
MTA	Message Transport Agent
NCSA	National Center for Supercomputing Applications
NPL	Netscape Public License
o.ä.	oder ähnliche
o.g.	oben genannt
OS	Open Source
OSI	Open Source Initiative
OSS	Open Source Software
PD	Public Domain
S.	Seite
sog.	sogenannt
Tab.	Tabelle
vgl.	vergleiche
u.a.	unter anderem
überarb.	überarbeitet
UrhG	Urheberrechtsgesetz
u.U.	unter Umständen
verb.	verbessert
z.B.	zum Beispiel
z.T.	zum Teil



# Kapitel 1

## Einleitung

*Open Source* bedeutet wörtlich übersetzt *offene Quelle*. Im Zusammenhang mit Software ist hiermit üblicherweise die freie, uneingeschränkte Verfügbarkeit, Nutzung und Modifikation des Softwarequellcodes gemeint<sup>1</sup>.

Obwohl Open Source Software (OSS) erst in jüngster Zeit von einer breiteren Öffentlichkeit wahrgenommen wird, lassen sich ihre Ursprünge bis in die 60-er Jahre zurückverfolgen. Einige der bedeutendsten Fortschritte in der Computertechnik basieren auf den Leistungen einer bereits nahezu in Vergessenheit geratenen *Hackerkultur*. Kapitel zwei geht in diesem Zusammenhang der Frage nach, welche geschichtlichen Ereignisse zur Entstehung freier Software beigetragen haben und welche Verbreitung sie im Rahmen der heutigen Softwarenutzung erreichen konnte. Zuvor soll jedoch erörtert werden, was sich im einzelnen hinter den Begriffen *Open Source*<sup>2</sup> und *freie Software* verbirgt. Schließlich werden im folgenden Kapitel verschiedene OSS-Lizenzen vorgestellt und eine Abgrenzung freier Software gegenüber anderen, in diesem Zusammenhang oftmals genannten Softwarekonzepten vorgenommen.

Nachdem zunächst grundlegende Informationen über Open Source Software vermittelt wurden, beleuchtet Kapitel drei die Art und Weise, in der derartige Software entsteht. Einem Überblick über allgemeine Aspekte folgt die Darstellung der Organisation freier Softwareprojekte sowie ihrer DV-technischen Unterstützung. Abschlie-

---

<sup>1</sup> Vgl. Nüttgens/Tesei (2001a), S. 2. Detaillierte Bestimmungen hinsichtlich Nutzung und Modifikation des Quellcodes finden sich in der zugehörigen Softwarelizenz, d.h. in den von ihrem Urheber eingeräumten Nutzungsrechten an der Software. Unter *Quellcode* werden die für Menschen lesbaren Anweisungen an einen Computer, welche in einer bestimmten (höheren) Programmiersprache formuliert wurden, verstanden. Ergebnis der Übersetzung des Quellcodes durch einen Compiler ist der maschinenlesbare Objektcode, vgl. Bundesministerium für Wirtschaft und Technologie (BMWi) (Hrsg.) (2001), S. 8 und S. 47.

<sup>2</sup> Die Begriffe *Open Source* und *Open Source Software* werden in dieser Arbeit synonym zueinander verwendet. Neben den eigentlichen Computerprogrammen wird in der Literatur unter *Software* zunehmend auch deren Dokumentation verstanden (vgl. bspw. Schwander (1996), S. 24 - 29.). Da jedoch eine solche Differenzierung für die vorliegende Arbeit unerheblich ist, wird der Begriff *Software* im folgenden gleichbedeutend zu *Programm* genutzt.

ßend wird in diesem Kapitel modellartig der typische Verlauf von OSS-Projekten vorgestellt.

Die Entwicklung freier Software hat ihren Ursprung außerhalb der klassischen Umgebungen von Produktion und Markt. Informations- und Kommunikationstechniken haben zudem völlig neue Voraussetzungen für wirtschaftliches Handeln mit sich gebracht. Kapitel vier beleuchtet in diesem Zusammenhang die Möglichkeit, inwieweit die Konzepte einer verteilten Produktion und einer freien Distribution die ökonomischen Modelle, welche gegenwärtig die Geschäftspolitik in der Softwareindustrie bestimmen, beeinflussen können. Es wird untersucht, auf welche Arten Unternehmen von einem auf freier Software basierenden Geschäftsmodell, dessen wesentliche Grundlage uneingeschränkt genutzt und verändert sowie jederzeit zu nur unbedeutenden Kosten reproduziert und weitergegeben werden darf, profitieren können. Darüber hinaus sollen in diesem Kapitel Risiken für verschiedene Unternehmen sowie mögliche neue Konzepte aufgrund der zunehmenden Verbreitung von OSS dargestellt werden. Schließlich wird der Versuch unternommen, grundlegende Ansätze und Methoden einer offenen, verteilten und kooperativen Form der Zusammenarbeit auf die Organisation wirtschaftlichen Handelns zu übertragen.

Kapitel fünf geht schließlich der Frage nach, inwiefern das Open Source Software-Modell sinnvoll auf (digitale) Daten im Allgemeinen angewendet werden kann. Im einzelnen wird untersucht, ob eine technische, organisatorische und rechtliche Übertragbarkeit der dieses Modell kennzeichnenden Merkmale auf Daten und Projekte jenseits der Softwareentwicklung möglich ist.

Beiträge und Veröffentlichungen über Open Source Software beschränken sich noch immer häufig auf GNU/Linux. Hinter dieser Art Software verbirgt sich jedoch mehr als ein kostenloses Betriebssystem. Sowohl Motivation als auch Ziel dieser Arbeit ist es, das *Phänomen* Open Source Software, welches bislang nur selten Gegenstand (wissenschaftlicher) Publikationen gewesen ist, in den genannten Aspekten näher zu beleuchten und weitere Untersuchungen hierüber anzuregen.

# Kapitel 2

## Grundlagen

### 2.1 Open Source – Definition, Begriffe, Abgrenzung und Lizenzen

#### 2.1.1 The Open Source Definition

Mit dem Begriff *Open Source* bzw. *Open Source Software* ist keine allgemein gültige, *offizielle* Definition verbunden, von zahlreichen Personen und Institutionen liegen ebenso viele individuelle Begriffsbestimmungen vor.

Für diese Arbeit soll diejenige der Open Source Initiative (OSI)<sup>1</sup> übernommen werden. Im Vergleich zu anderen Definitionen stellt sie über die Forderungen nach uneingeschränkter Verfügbarkeit und möglichen Modifikationen des Quellcodes hinaus weitere Bedingungen an eine Softwarelizenz, bei deren Erfüllung sich ein Programm, das unter der betrachteten Lizenz veröffentlicht wird, die Bezeichnung *OSI Certified Open Source Software* erhält<sup>2</sup>. Die OSI prägte den Begriff *Open Source*, um einerseits eine Tradition öffentlich zugänglichen Quellcodes und gemeinschaftlicher Softwareentwicklung zu beschreiben und um andererseits eine gemeinsame Bezeichnung für diejenigen Softwarelizenzen nutzen zu können, welche nach ihrer Überzeugung “Freiheit im Sinne der Wissenschaft<sup>3</sup>” verkörpern.

Die Definition der Open Source Initiative gilt als vergleichsweise umfassend und ist weitgehend anerkannt. Im einzelnen umfasst *The Open Source Definition* die folgenden neun Bedingungen<sup>4</sup>:

1. Freie Weiterverbreitung: Die Lizenz darf niemanden am Verkauf oder an der unentgeltlichen Weitergabe der Software hindern. Lizenzgebühren oder vergleich-

---

<sup>1</sup> Vgl. Abschnitt 2.2.

<sup>2</sup> Bei genauerer Betrachtung ist die *Open Source Definition* der OSI eine Kombination aus einer Definition und Rahmenbedingungen für Softwarelizenzen.

<sup>3</sup> Müller (1999), S. 8.

<sup>4</sup> Vgl. Open Source Initiative (Hrsg.) (2001).

bare Formen der Vergütung sind jedoch nicht zulässig.

2. Quellcode: Der Quellcode muss der Software beiliegen. Sofern die Software in kompilierter Form vertrieben wird, muss eine öffentlich zugängliche und gebührenfreie Bezugsquelle für den Quellcode genannt werden.
3. Abgeleitete Werke: Die Lizenz muss Veränderungen am ursprünglichen Programm erlauben sowie die Möglichkeit, Programme basierend auf dem ursprünglichen Programm zu erstellen. Diese sog. *abgeleiteten Werke* müssen unter denselben Lizenzbestimmungen wie das ursprüngliche Programm veröffentlicht werden können, abweichende Lizenzen sind jedoch ebenfalls zulässig.
4. Unversehrtheit des Originalquellcodes: Die Verbreitung modifizierten Quellcodes kann nur dann untersagt werden, wenn die Modifikationen in zusätzlichen Dateien<sup>5</sup> zusammen mit dem Originalquellcode veröffentlicht werden dürfen<sup>6</sup>. Zudem können die Lizenzbestimmungen von abgeleiteten Werken verlangen, dass sie eine vom ursprünglichen Programm abweichende Bezeichnung oder Versionsnummer erhalten.
5. Keine Diskriminierung von Personen oder Gruppen: Personen oder Personengruppen dürfen von der Lizenz nicht unterschiedlich behandelt werden.
6. Keine Beschränkung auf bestimmte Anwendungsgebiete: Die Lizenz darf niemanden an der Benutzung der Software in einem bestimmten Bereich hindern. Das Programm muss auch im kommerziellen Bereich eingesetzt werden dürfen.
7. Verbreitung der Lizenz: Die zum Programm gehörenden Rechte müssen in ihrer ursprünglichen Form für jeden gelten, der das Programm erhalten hat. Zusätzliche Bestimmungen durch weitere Lizenzen oder Vereinbarungen (z.B. durch *Non-Disclosure Agreements*) sind nicht zulässig.
8. Keine Beschränkung auf ein bestimmtes Produkt: Wird das Programm außerhalb einer Softwaredistribution genutzt oder von dieser getrennt verbreitet, so müssen für das Programm dieselben Rechte gelten, wie sie für die ursprüngliche Distribution gewährt wurden.
9. Keine Beeinträchtigung anderer Software: Die Lizenzbestimmungen anderer Programme, die mit der lizenzierten Software gemeinsam verbreitet werden, dürfen in keiner Weise beeinflusst werden.

---

<sup>5</sup> Gemeint sind sog. *Patches*. Ein Patch (Flicken) ist ein Stück Software, das durch Integration in ein zugehöriges Programm Fehler in diesem behebt oder dessen Funktionalität erweitert.

<sup>6</sup> Die Q Public License des norwegischen Unternehmens Trolltech beinhaltet bspw. eine solche Bestimmung, vgl. Trolltech (Hrsg.) (2001).

Softwarelizenzen, die von der OSI hinsichtlich der Erfüllung dieser Punkte überprüft wurden und bei denen eine Übereinstimmung mit den genannten Forderungen festgestellt werden konnte, sind z.B. die Mozilla Public License, die IBM Public License, die GNU General Public License sowie die GNU Lesser General Public License<sup>7</sup>.

Die Free Software Foundation (FSF)<sup>8</sup> um Richard M. Stallman<sup>9</sup> dagegen bevorzugt aufgrund ihrer abweichenden Auffassung über *freie* Software die Bezeichnung *Free Software* im Gegensatz zu *Open Source Software*. Ihrer Ansicht nach kommt der Gedanke der freien Nutzung und Veränderung von Software in der Open Source Definition zu kurz, eine Kommerzialisierung freier Software sei zu befürchten. Vor allem die Möglichkeit, abgeleitete Werke unter abweichenden Lizenzbestimmungen zu veröffentlichen, widerspricht den Vorstellungen der FSF hinsichtlich der unbedingten Weitergabe modifizierter Software als *freie* Software<sup>10</sup>.

Obwohl sowohl im englischen als auch im deutschen Sprachgebrauch häufig *Free Software* bzw. *freie Software* synonym zu *Open Source Software* verwendet wird, bestehen zwischen den Initiatoren dieser Begriffe teilweise abweichende Auffassungen über deren Bedeutung<sup>11</sup>.

Beide Organisationen weisen jedoch ausdrücklich darauf hin, dass Free Software bzw. Open Source Software nicht unbedingt kostenlose Software meint, sondern vielmehr die persönliche Freiheit des Anwenders, die Software nach seinen Wünschen einsetzen und verändern zu können. Eine Beschränkung der eingeräumten Rechte auf einen bestimmten Anwenderkreis oder auf einen bestimmten Anwendungsbereich ist nicht zulässig<sup>12</sup>.

Zusammenfassend kann festgehalten werden, dass *Free Software* *Open Source Software* ist, die umgekehrte Aussage gilt jedoch nicht in jedem Fall. Open Source Software schließt Free Software im Sinne der FSF ein. Da eine solche Abgrenzung jedoch nur schwer aufrechtzuerhalten ist, wird in dieser Arbeit der Begriff *freie Software* gleichbedeutend zu *Open Source Software* (OSS) genutzt.

---

<sup>7</sup> Vgl. Abschnitt 2.1.4.

<sup>8</sup> Vgl. Abschnitt 2.2.

<sup>9</sup> Stallman ist einer der Protagonisten der Free Software-Gemeinschaft und Initiator der Free Software Foundation. Vgl. Stallman (2001a).

<sup>10</sup> Vgl. Free Software Foundation (Hrsg.) (2001a).

<sup>11</sup> "The core of the GNU project is the idea of free software as a social, ethical, political issue. ... Free software is a political philosophy and open source is a development methodology." Stallman (2001b). Vgl. bzgl. des GNU-Projekts Abschnitt 2.2.

<sup>12</sup> Vgl. Free Software Foundation (Hrsg.) (2001b).

### 2.1.2 Ergänzende Begriffsbestimmungen

Ebensowenig wie für *Open Source Software* eine allgemein gültige Definition vorliegt, lassen sich für die folgenden, im Zusammenhang mit freier Software relevanten Begriffe keine generellen Definitionen geben. Aus diesem Grund soll an dieser Stelle lediglich eine Umschreibung erfolgen.

#### 2.1.2.1 Copyleft

Richard M. Stallman vertritt die Überzeugung, dass die freie Verfügbarkeit des Softwarequellcodes für Weiterentwicklungen und Innovationen auf dem Gebiet der Computerwissenschaft notwendig sei<sup>13</sup>. Copyleft ist ein vom ihm entwickeltes Konzept, das jedem Nutzer die Freiheit geben soll, Software nach seinen Vorstellungen einzusetzen, zu verändern und weiterzuverteilen. Jede Weitergabe von Copyleft-Software – gleichgültig, ob sie modifiziert wurde oder nicht – muss wiederum gewährleisten, dass für alle Nutzer dieselben Freiheiten gelten, wie sie für diejenigen der ursprünglichen Software eingeräumt wurden. Eine Beschränkung der Rechte durch zusätzliche Reglementierungen ist nicht erlaubt<sup>14</sup>. Die konkrete, juristisch verbindliche Umsetzung des Copyleft-Konzepts erfolgt in den entsprechenden Softwarelizenzen, wie z.B. der GNU General Public License.

#### 2.1.2.2 Hacker

Durch die neue, *interaktive* Form der Computernutzung mittels Time Sharing-Systemen lösten sich Programmierer in den 60-er Jahren vereinzelt von der althergebrachten Methode der Softwareentwicklung, welche größtenteils abstrakt auf Papier stattfand. Es war nun vielmehr möglich, eine Idee per Tastatur einzugeben, das Programm laufen zu lassen, Fehler zu entdecken, die Korrekturen vorzunehmen und es sofort wieder ablaufen zu lassen. Diese Art der iterativen Programmierung trug den Namen *Hacken*<sup>15</sup>.

Die Wurzeln freier Software sind untrennbar mit denen der Hackergemeinschaften, die sich vor allem an Universitäten anfangs der 60-er Jahre bildeten, verbunden. Sie waren die ersten, die freie Software entwickelten, obwohl es den Begriff zu dieser Zeit noch gar nicht gab<sup>16</sup>.

<sup>13</sup> Vgl. DiBona/Ockman/Stone (1999), S. 2.

<sup>14</sup> Vgl. Free Software Foundation (Hrsg.) (2001c) und Lerner/Tirole (2000), S. 5 f. "Copyleft uses copyright law, but flips it over to serve the opposite of its usual purpose: instead of a means of privatizing software, it becomes a means of keeping software free." Stallman (1999), S. 59.

<sup>15</sup> Vgl. Levy (1994), S. 21 f.

<sup>16</sup> Vgl. Raymond (2001b), S. 4 - 7. Einen Überblick über Normen und Verhaltensweisen innerhalb der Hackerkultur liefert Eric S. Raymond in seinem Essay *Homesteading the Noosphere*, vgl. Raymond (2001c).

Durch abweichende Darstellungen in den Medien wurden mit der Zeit Hacker als Personen bezeichnet, die böswillig in fremde Computersysteme einbrechen. Bei derartigen Personen handelt es sich jedoch nicht um Hacker, sondern um *Cracker*: “The basic difference is this: hackers build things, crackers break them<sup>17</sup>.”

### 2.1.2.3 Community

*Community* bedeutet in etwa *Gemeinschaft* oder *Gemeinde*. Bei Open Source-Communities handelt es sich um freie Interessengemeinschaften verschiedener Personen, welche sich über entsprechende Kommunikationsmittel<sup>18</sup> organisieren und gemeinsam an einem Projekt arbeiten. Die Anfänge freier Software gehen auf derartige Communities zurück und noch immer entsteht OSS vor allem in und durch solche Gemeinschaften.

Bei genauerer Betrachtung besteht die Open Source-Community aus vielen themen- oder interessensspezifischen Teil-Communities, die in ihrer Gesamtheit ein verbundenes und untereinander verflochtenes System von Gruppen und Projekten darstellen. Diese Communities unterscheiden sich nicht nur in der Größe ihres Projekts und in der ohnehin schwankenden Anzahl ihrer Mitglieder, sondern u.a. in der von ihnen verfolgten Lizenzpolitik und in dem Ausmaß ihrer Kooperation mit kommerziellen Interessengruppen<sup>19</sup>.

## 2.1.3 Abgrenzung freier Software von nicht-freier Software

Zu Missverständnissen führt oftmals die Tatsache, dass im Zusammenhang mit freier Software gleichbedeutend andere, nicht-freie Softwarekategorien bzw. -konzepte genannt werden. Nachfolgend sollen die gebräuchlichsten derartigen Kategorien vorgestellt und eine Abgrenzung zu freier Software vorgenommen werden.

### 2.1.3.1 Public Domain-Software

Public Domain-Software<sup>20</sup> ist eine US-amerikanische Besonderheit. Es kann sich hierbei um Software handeln, die entweder durch staatliche Unterstützung – z.B. an Universitäten – entstanden und somit aufgrund gesetzlicher Bestimmungen kostenlos sowie für die Öffentlichkeit uneingeschränkt zugänglich ist oder um solche, deren

---

<sup>17</sup> Raymond (2001a), S. 196.

<sup>18</sup> Vgl. Abschnitt 3.3.1.

<sup>19</sup> Vgl. Nüttgens/Tesei (2001a), S. 10 - 18 und Raymond (2001c), S. 67 - 71.

<sup>20</sup> *Public Domain* lässt sich am ehesten mit *Allgemeingut* oder mit *Gemeinfreiheit* übersetzen, vgl. Müller (1999), S. 9.

Autoren gänzlich auf ihr Copyright verzichten<sup>21</sup>.

Derartige Software unterliegt keinen urheberrechtlichen Bestimmungen, sie kann uneingeschränkt und vergütungsfrei genutzt, modifiziert und verbreitet werden<sup>22</sup>. Bei Public Domain-Software handelt es sich um freie Software.

### 2.1.3.2 Shareware

Shareware ist eine Form der Softwaredistribution, welche es dem Anwender zu Testzwecken erlaubt, die frei zugängliche, i.d.R. jedoch nur in kompilierter Form veröffentlichte Software für einen bestimmten, vom Autor festgelegten und in den Lizenzbestimmungen genannten Zeitraum kostenlos zu nutzen. Will der Anwender die Software nach Ablauf dieses sog. Testzeitraums weiterhin nutzen, so muss er eine Lizenzgebühr an den Autor entrichten, andernfalls liegt ein Verstoß gegen das Urheberrecht vor<sup>23</sup>. Shareware kann im Allgemeinen uneingeschränkt weiterverteilt werden, Modifikationen sind jedoch – sofern der Quellcode überhaupt zugänglich ist – untersagt<sup>24</sup>. Derartige Programme sind demnach nicht-freie Software.

### 2.1.3.3 Freeware und Semi-Free Software

Bei Freeware handelt es sich um die lizenzgebührenfreie Entsprechung von Shareware, Änderungen an der Software sind demnach ebensowenig erlaubt<sup>25</sup>. Schränkt der Autor die lizenzgebührenfreie Nutzung auf einen bestimmten Anwenderkreis (z.B. Privatanwender) ein, so handelt es sich um Semi-Free Software<sup>26</sup>. Derartige Software liegt i.d.R. lediglich in kompilierter Form vor. Weder Freeware noch Semi-Free Software sind freie Software.

### 2.1.3.4 Proprietäre Software

Sowohl der Gebrauch als auch die Veränderung und die Weitergabe proprietärer Software unterliegen der Erlaubnis ihres Urhebers<sup>27</sup>. Proprietäre Software wird von Personen oder Unternehmen unter Geheimhaltung des Quellcodes entwickelt und veröffent-

<sup>21</sup> Vgl. Nüttgens/Tesei (2001b), S. 14. Nach §201, Absatz d, Ziffer 1 des *Copyright Law of the United States of America* kann ein Autor auf sein individuelles Copyright verzichten. Das deutsche Urheberrechtsgesetz (UrhG) sieht eine solche Möglichkeit dagegen nicht vor. Vgl. The Library of Congress (Hrsg.) (2001).

<sup>22</sup> Aufgrund des Fehlens urheber- bzw. lizenzrechtlicher Bestimmungen kann Public Domain-Software jederzeit Grundlage der Entwicklung proprietärer Software sein. Diese Möglichkeit hat letztendlich wesentlich zur Formulierung von OSS-Lizenzen beigetragen. Vgl. zu proprietärer Software Abschnitt 2.1.3.4.

<sup>23</sup> Vgl. Nüttgens/Tesei (2001b), S. 14 ff.

<sup>24</sup> Vgl. Lerner/Tirole (2001), S. 821.

<sup>25</sup> Vgl. Nüttgens/Tesei (2001b), S. 16.

<sup>26</sup> Vgl. Free Software Foundation (Hrsg.) (2001d).

<sup>27</sup> Vgl. Nüttgens/Tesei (2001a), S. 2.



Software-kategorie	Quellcode verfügbar und modifizierbar	Modifikationen unterliegen stets denselben Bestimmungen	dauerhaft lizenzgebührenfrei	uneingeschränkte Nutzung	Weiterverteilung erlaubt
OSS	X	(X)	X	X	X
PD	X		X	X	X
Freeware			X	X	X
Semi-Free			X		X
Shareware					X
Proprietär					

Tab. 2.1.3/1: Vergleich verschiedener Softwarekategorien

licht<sup>28</sup> sowie gegen Lizenzgebühren zum Kauf angeboten. Bei derartiger Software handelt es sich demnach um das Gegenteil von freier Software<sup>29</sup>.

Tabelle 2.1.3/1 veranschaulicht die wesentlichen Merkmale der genannten Softwarekategorien im Unterschied zu Open Source Software (OSS).

## 2.1.4 Softwarelizenzen

Die Lizenzierung von Software ist nicht nur im kommerziellen Bereich zu beobachten. Richard M. Stallman erkannte schon frühzeitig, dass rechtliche Rahmenbedingungen durch juristisch verbindliche Lizenzbestimmungen für den Erhalt freier Software unabdingbar sind. Von verschiedenen Personen und Gruppen aus der Open Source-Community wurden – nicht zuletzt durch das zunehmende Interesse kommerzieller Softwareunternehmen – mit der Zeit zahlreiche Softwarelizenzen<sup>30</sup> formuliert, die mitunter in ihren Inhalten nur geringfügig voneinander abweichen, in ihrer Verbreitung jedoch wesentliche Unterschiede aufweisen. An dieser Stelle sollen die gebräuchlichsten OSS-Lizenzen überblicksartig vorgestellt werden.

<sup>28</sup> Proprietäre Software wird daher mitunter als *Closed Source Software* bezeichnet.

<sup>29</sup> Die Begriffe *kommerzielle Software* und *proprietäre Software* werden oft synonym zueinander verwendet. Da proprietäre Software i.d.R. kommerziell verwertet wird, ist in dieser Arbeit mit *kommerzieller Software* stets *proprietäre Software* gemeint.

<sup>30</sup> Allein die Anzahl der Lizenzen, die von der OSI als übereinstimmend mit der Open Source Definition erklärt werden, beläuft sich auf 26. Stand: 2001-09-28.

### 2.1.4.1 GNU General Public License (GPL)

Die GPL ist vermutlich die bekannteste und gebräuchlichste Lizenz freier Software<sup>31</sup>. Sie erlaubt es, die ihr unterliegenden Programme uneingeschränkt zu nutzen, zu verändern, zu kopieren und weiterzugeben<sup>32</sup>. Software, die unter diese Lizenz gestellt wird, muss im Quellcode vorliegen. Durch Umsetzung des Copyleft-Konzepts verlangt sie zudem von einem Programm, welches ganz oder teilweise auf einem anderen GPL-Programm basiert – d.h. von einem abgeleiteten Werk – dass jenes ebenfalls den Bestimmungen der GPL zu unterliegen hat<sup>33</sup>. Sie verhindert somit, dass Software, die unter dieser Lizenz veröffentlicht wird, in proprietäre Software umgewandelt werden kann<sup>34</sup>.

### 2.1.4.2 GNU Lesser General Public License (LGPL)

Die ebenfalls von der FSF formulierte GNU Lesser General Public License ist der Nachfolger der GNU Library General Public License und findet vorwiegend auf Softwarebibliotheken Anwendung. Der wesentliche Unterschied zur GPL ist, dass ein Programm, welches auf einer LGPL-Bibliothek basiert, auch nicht-frei sein darf, indem es nicht als ein abgeleitetes Werk im Sinne der GPL betrachtet wird und beliebigen Lizenzen unterstellt werden kann. Änderungen an der Bibliothek selbst müssen jedoch wieder der LGPL (oder der GPL) unterliegen<sup>35</sup>. Während Software, die Funktionalitäten von GPL-Bibliotheken nutzt, ebenfalls der GPL zu unterliegen hat, ermöglicht die LGPL die kommerzielle Verwertung von auf LGPL-Bibliotheken beruhender Software. Ein wesentlicher Grund für die Formulierung der LGPL war die Erhöhung des Verbreitungsgrades freier Software<sup>36</sup>.

### 2.1.4.3 Sonstige Lizenzen

Neben den oben genannten gibt es noch zahlreiche andere Softwarelizenzen im Open Source-Umfeld, deren Beschreibung an dieser Stelle jedoch zu weit führen würde. Aufgrund ihrer Bekanntheit seien lediglich die Mozilla Public License der Mozilla

<sup>31</sup> In einer Untersuchung kommt David A. Wheeler zu dem Ergebnis, dass ca. 50% des Quellcodes der GNU/Linux Distribution Red Hat Linux 7.1 der GPL unterliegt, vgl. Wheeler (2001).

<sup>32</sup> Die Verbreitung der Software kann auch gegen eine Gebühr erfolgen. Hierbei handelt es sich jedoch nicht um eine Lizenzgebühr, sondern vielmehr um einen Ausgleich für den durch das Anfertigen und Verteilen der Kopien entstandenen Aufwand. Vgl. Nüttgens/Tesei (2001a), S. 20 f. Die wirtschaftliche Verwertung freier Software sollte durch diese Gebühr nicht gänzlich ausgeschlossen werden.

<sup>33</sup> Vgl. Free Software Foundation (Hrsg.) (2001e). Wegen dieser zentralen, auf Unternehmen der Softwareindustrie häufig abschreckend wirkenden Lizenzbestimmung erhielt die GPL von ihren Kritikern die Bezeichnung *GNU Public Virus*.

<sup>34</sup> Vgl. zur Konformität der GPL mit dem UrhG Abschnitt 5.3.1.

<sup>35</sup> Vgl. Free Software Foundation (Hrsg.) (2001f).

<sup>36</sup> Vgl. Müller (1999), S. 10.

Organisation<sup>37</sup> und die Apache Software License der Apache Software Foundation<sup>38</sup> erwähnt. Derartige Lizenzen – welche teilweise auf der GPL bzw. der LGPL basieren – verfolgen im wesentlichen das Ziel, sowohl Programmierern als auch Anwendern “Software mit möglichst wenigen Einschränkungen zur Verfügung zu stellen und gleichzeitig ihren Fortbestand und ihre Weiterentwicklung zu sichern oder zumindest zu vereinfachen<sup>39</sup>.”

## **2.2 Historische Entwicklung freier Software und der Open Source Software-Bewegung**

Von entscheidender Bedeutung für die historische Entwicklung freier Software ist das Betriebssystem UNIX, welches nicht zuletzt durch seinen Verbreitungsgrad und seine zunächst weitgehend freie Verfügbarkeit wesentlich zur Entstehung dieser Art Software beigetragen hat. Darüber hinaus sollen im folgenden weitere grundlegende Ereignisse, die zur Entstehung und Verbreitung freier Software beigetragen haben, vorgestellt werden.

### **Die ersten Software-Sharing Communities**

Die kommerzielle Entwicklung von Software spielte in den Anfängen der Computerindustrie nur eine unwesentliche Rolle. Das Computergeschäft bestand zu dieser Zeit meist nur in dem Verkauf von Hardware. Da die meisten Anwender ihre Programme selbst schrieben, war Software damals nicht mehr als ein Nebenprodukt der Hardware<sup>40</sup>.

Mit der Verfügbarkeit von Time Sharing-Systemen als Alternative zu Batch-Systemen in den 60-er Jahren jedoch gewann die Softwareentwicklung an Bedeutung. Festplatten und Magnetbänder ermöglichten darüber hinaus eine Speicherung und Modifikation von Programmen auf dem Rechner selbst. Diese neue Form der Computernutzung sowie die damals selbstverständliche Offenlegung des Quellcodes förderten die Kooperation der Nutzer sowie den gegenseitigen Austausch von Programmen<sup>41</sup>.

Richard M. Stallman prägte für diese Gemeinschaften von Computernutzern, die den Quellcode ihrer Programme offenlegten, untereinander Software austauschten und sich gegenseitig bei der Verbesserung ihrer Programme halfen, den Begriff der

---

<sup>37</sup> Vgl. The Mozilla Organisation (Hrsg.) (2001a).

<sup>38</sup> Vgl. The Apache Software Foundation (Hrsg.) (2001a).

<sup>39</sup> Müller (1999), S. 9.

<sup>40</sup> Vgl. Müller (1999), S. 16.

<sup>41</sup> Vgl. Nüttgens (2001), S. 3 f. Zudem fiel Software zu dieser Zeit noch nicht unter die durch das Urheberrecht schutzfähigen Werke. In den USA empfahl die *National Commission on New Technological Uses of Copyrighted Works* (CONTU) erstmals 1976, Computerprogramme unter den Schutz des Copyright Law zu stellen. Vgl. Borchers (2001).

*Software-Sharing Community*<sup>42</sup>.

## Die Entwicklung von UNIX

1969 begann Ken Thompson in den AT&T Bell Telephone Laboratories mit der Entwicklung von UNIX als Ersatz für das gescheiterte Betriebssystem MULTICS<sup>43</sup>. Parallel arbeitete dort Dennis Ritchie an der Entwicklung der Programmiersprache C. 1971 beschlossen Thompson und Ritchie, das bislang in Assembler auf den damals verbreiteten Rechnern der PDP-Serie von Digital Equipment Corporation (DEC) geschriebene UNIX in C neu zu schreiben. Auf der Basis dieser Zusammenarbeit wurde im November 1973 mit UNIX, 4th Edition, erstmals ein Betriebssystem vorgestellt, das auf verschiedenartige Rechner portiert werden konnte<sup>44</sup>.

AT&T blieb jedoch aufgrund einer Vereinbarung mit dem US-Justizministerium aus dem Jahr 1956 der Einstieg in die Computerbranche untersagt, weshalb UNIX gegen nur geringe Lizenzgebühren<sup>45</sup> an Universitäten und andere Institutionen abgegeben werden konnte. Aufgrund der mit C verbundenen Portierbarkeit und Zugänglichkeit sowie der geringen Lizenzgebühren fand UNIX vor allem im universitären Umfeld rasch Verbreitung<sup>46</sup>.

Zudem erlaubte das bis zu diesem Zeitpunkt entwickelte ARPAnet eine wesentliche Erleichterung und Beschleunigung des Informationsaustausches zwischen den angebundenen Institutionen. Da von AT&T nur marginale Unterstützung und Beratung geboten wurde, entwickelte sich mit der Zeit zwischen den Universitäten ein schnelles und leistungsfähiges UNIX-Support-Netzwerk<sup>47</sup>, wodurch das Betriebssystem bald einen hohen Reifegrad erreichen konnte<sup>48</sup>.

<sup>42</sup> Vgl. Stallman (1999), S. 53 und Lerner/Tirole (2000), S. 4.

<sup>43</sup> MULTICS steht für *MULTiplexed Information and Computing Service*. UNIX sollte sich durch *Einfachheit* von MULTICS unterscheiden, weshalb es den Namen *UNiplexed Information and Computing Service* erhielt, aus dessen Abkürzung UNICS schließlich *UNIX* wurde. Vgl. Tanenbaum (1987), S. 10 f.

<sup>44</sup> Vgl. Raymond (2001b), S. 22 f. und Salus (1994), S. 43. Unter *Portierung* wird die Anpassung einer Software an bzw. ihre Übertragung auf eine neue oder geänderte Hardwareumgebung verstanden. Vgl. Schwander (1996), S. 21.

<sup>45</sup> Anfangs beliefen sich die Lizenzgebühren für UNIX einschließlich des Quellcodes auf \$50, vgl. Grassmuck (2001a).

<sup>46</sup> Vgl. Duncan/Hull (2001).

<sup>47</sup> Als zentrale Koordinationsstelle fungierte schon bald die Universität von Kalifornien in Berkeley. Der spätere Gründer von Sun Microsystems, William Joy, sammelte dort alle Korrekturen und Ergänzungen, arbeitete selbst an zahlreichen Erweiterungen von UNIX – bspw. an der durch die Advanced Research Projects Agency (ARPA) finanzierten Integration von TCP/IP – mit und veröffentlichte diese erstmals 1978 als Berkeley Software Distribution (BSD UNIX). Vgl. Grassmuck (2001b).

<sup>48</sup> Vgl. Müller (1999), S. 16. „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.“ Gulbins/Obermayr (1995), S. 7.

### Das GNU-Projekt und die Free Software Foundation

1971 wurde Richard M. Stallman Mitglied der Software-Sharing Community im Artificial Intelligence Laboratory des Massachusetts Institute of Technology (MIT), das in den 70-er Jahren eine der einflussreichsten Hacker-Communities beherbergte. Als jedoch Unternehmen der Computerindustrie zunehmend erkannten, dass sich mit Software erhebliche Erlöse erzielen lassen und immer mehr Hacker von diesen Unternehmen abgeworben wurden, löste sich die Hacker-Community am MIT anfangs der 80-er Jahre zunehmend auf<sup>49</sup>.

Zur selben Zeit veröffentlichten Unternehmen wie IBM, Hewlett-Packard und DEC die ersten kommerziellen UNIX-Varianten, welche nur auf der Hardware von Rechnern ihres Unternehmens lauffähig waren. AT&T selbst wurde aufgrund einer Entscheidung des US-Justizministeriums mit Wirkung zum 1. Januar 1984 in mehrere Unternehmen aufgeteilt und durfte im Gegenzug von nun an als Wettbewerber in der Computerindustrie auftreten<sup>50</sup>. Zu diesem Zweck gründete AT&T das Tochterunternehmen UNIX System Laboratories (USL). Durch die Möglichkeit der kommerziellen Nutzung hob USL die Lizenzgebühren für UNIX drastisch an<sup>51</sup> und setzte somit dem Zugriff auf den Quellcode, an dem bislang zahllose Programmierer in der ganzen Welt mitgearbeitet hatten, sowie dem Austausch von UNIX-Programmen faktisch ein Ende<sup>52</sup>.

Im Zuge dieser Veränderungen kündigte Stallman 1984 seine Stelle am MIT und gründete das GNU-Projekt<sup>53</sup> mit dem Ziel, ein freies, zu UNIX funktional äquivalentes Betriebssystem in der Programmiersprache C zu entwickeln, das die von ihm entschieden abgelehnte proprietäre Software irgendwann einmal überflüssig machen solle<sup>54</sup>. Erste Ergebnisse des GNU-Projekts waren der GNU C Compiler gcc<sup>55</sup> und der Editor GNU Emacs. Um zu verhindern, dass GNU-Software in proprietäre Software umgewandelt werden kann, entwickelte Stallman das Copyleft-Konzept. Ein Jahr später gründete er die Free Software Foundation (FSF), eine gemeinnützige Organisation als Koordinationsstelle und zur Erwirtschaftung von Einnahmen durch den Versand von Kopien der im Rahmen des GNU-Projekts erstellten Software und

<sup>49</sup> Vgl. Stallman (1999), S. 53 f. und BMWi (Hrsg.) (2001), S. 8 f.

<sup>50</sup> Vgl. Duncan/Hull (2001).

<sup>51</sup> Die Lizenzgebühren für UNIX stiegen mit der Zeit auf bis zu \$100.000, vgl. Grassmuck (2001a).

<sup>52</sup> Vgl. Müller (1999), S. 16.

<sup>53</sup> GNU ist ein rekursives Akronym für *GNU's Not UNIX*. Die Tatsache, dass Stallman am renommierten MIT beschäftigt war, mag dazu beigetragen haben, dass ihm und seinem Projekt die für einen Erfolg notwendige Glaubwürdigkeit und Beachtung zu Teil wurden.

<sup>54</sup> Vgl. Stallman (2001c). Seine endgültige Entscheidung, das GNU-Projekt zu gründen, beruht angeblich auf der Weigerung eines Händlers, Stallman den Quellcode eines Druckertreibers auszuhändigen, vgl. O'Reilly (2001).

<sup>55</sup> Der *GNU C Compiler* wurde im April 1999 nach der Zusammenführung mit dem Experimental GNU Compiler System egcs in *GNU Compiler Collection* umbenannt.

Dokumentationen<sup>56</sup>.

### GNU HURD und der Linux-Kernel

Mit der Zeit arbeiteten immer mehr Entwickler am GNU-Projekt mit. Nach und nach konnten UNIX-Komponenten durch entsprechende GNU-Programme ersetzt werden<sup>57</sup>. 1990 war das GNU-System nahezu komplett, die einzige, wesentliche fehlende Komponente war der Betriebssystem-Kernel<sup>58</sup>, den das GNU HURD-Projekt beisteuern sollte<sup>59</sup>. Die Entwicklung eines UNIX-kompatiblen Kernels erwies sich jedoch problematischer und langwieriger als erwartet. Die Fertigstellung von GNU HURD und damit des gesamten GNU-Systems sollte sich noch Jahre hinziehen<sup>60</sup>.

1991 jedoch entwickelte Linus Torvalds, ein Student der Informatik an der Universität Helsinki, einen zu UNIX funktional äquivalenten Kernel für Personal Computer mit Intel 80386-Mikroprozessoren, basierend auf dem UNIX-artigen Betriebssystem MINIX<sup>61</sup>. Ein Jahr später konnte dieser sog. Linux-Kernel, welchen Torvalds unter der GPL veröffentlichte, mit dem GNU-System kombiniert werden. Ergebnis dieser Zusammenführung ist das Betriebssystem GNU/Linux<sup>62</sup>. Mit der zunehmenden Akzeptanz und Nutzung des Internet Mitte der 90-er Jahre wurde GNU/Linux von einer breiteren Öffentlichkeit wahrgenommen. Die Aktivitäten interessierter Entwickler richteten sich verstärkt auf dieses Betriebssystem, welches zunehmend Stabilität erlangte und sich als Plattform für Entwickler etablieren konnte. Inzwischen gilt GNU/Linux als der Prototyp eines einheitlichen Systems für unterschiedliche Rechnerarchitekturen und einzige ernsthafte Alternative zu Microsoft Windows<sup>63</sup>.

### Die Open Source Initiative

Zu Beginn des Jahres 1998 fanden sich Eric S. Raymond<sup>64</sup> und andere Mitglieder der Community zusammen und diskutierten die Möglichkeiten, einen Begriff zu eta-

<sup>56</sup> Vgl. Müller (1999), S. 17.

<sup>57</sup> Die modulare Struktur von UNIX sollte sich für das Ersetzen seiner einzelnen Komponenten als eine wesentliche Erleichterung herausstellen. Vgl. Abschnitt 3.1.3.4.

<sup>58</sup> Der *Kernel* ist für elementare Funktionen des jeweiligen Betriebssystems zuständig und befindet sich zur Laufzeit stets im Arbeitsspeicher des Rechners. Zu seinen Hauptaufgaben zählen die Prozess- und die Betriebsmittelverwaltung. Vgl. Tanenbaum (1987), S. 87 - 92.

<sup>59</sup> HURD war damals bereits das zweite Projekt, dessen Ziel die Entwicklung eines Kernels war. *TRIX* scheiterte jedoch schon frühzeitig.

<sup>60</sup> Tatsächlich wurde die erste Release von GNU HURD 1996 vorgestellt.

<sup>61</sup> MINIX wurde ursprünglich von Andrew S. Tanenbaum zur Demonstration der Funktionalität und der Arbeitsweise eines Betriebssystems entwickelt. Vgl. Tanenbaum (1987), S. 13 f. Die Tatsache, dass Torvalds einen Großteil der MINIX-Community für Linux gewinnen konnte, mag wesentlich zu dessen Erfolg beigetragen haben.

<sup>62</sup> Vgl. Stallman (1999), S. 65 f. Oftmals wird nur von *Linux* gesprochen, richtiger aber ist die Bezeichnung *GNU/Linux*, da es sich bei Linux lediglich um den Betriebssystem-Kernel handelt.

<sup>63</sup> Vgl. Nüttgens/Tesei (2001a), S. 13 f.

<sup>64</sup> Raymond ist neben Richard M. Stallman einer der prominentesten Vertreter der Open Source-Community und eines der Gründungsmitglieder der OSI. Vgl. Raymond (2001e).

blieren, der die Anhänger freier Software der Notwendigkeit entheben kann, ständig sowohl zwischen kostenloser Software und freier Software als auch zwischen den verschiedenen Lizenzen derartiger Software unterscheiden zu müssen<sup>65</sup>.

Das Adjektiv *free* wurde durch *open* ersetzt, um nicht nur auf den Aspekt der kostenlosen Verfügbarkeit reduziert zu werden und um somit der verbreiteten Annahme entgegenzuwirken, die Befürworter eines derartigen Konzepts würden Software als öffentliches Gut betrachten und folglich keinen Personen oder Unternehmen Erlöse hieraus zugestehen<sup>66</sup>.

Ergebnis dieser Zusammenkunft ist die Gründung der Open Source Initiative (OSI), einer gemeinnützigen Organisation mit dem Ziel, den Open Source-Gedanken in den Bereich der kommerziellen Softwareentwicklung zu verbreiten und traditionelle Softwareunternehmen zur Freigabe des Quellcodes ihrer Produkte zu ermutigen. Kurz nach ihrer Gründung veröffentlichte die OSI ihre *Open Source Definition*<sup>67</sup>, die auf den *Debian Free Software Guidelines*<sup>68</sup> von Bruce Perens basiert<sup>69</sup>.

## 2.3 Ausgewählte Open Source Software-Projekte

Um einen Eindruck von der Bedeutung freier Software zu bekommen, soll der folgende Abschnitt einen Überblick über die neben GNU/Linux<sup>70</sup> verbreitetsten Open Source Software-Projekte vermitteln<sup>71</sup>.

### 2.3.1 Apache

Basis des Apache HTTP-Servers<sup>72</sup> ist der HTTP-Daemon<sup>73</sup> httpd, der ursprünglich von Rob McCool am National Center for Supercomputing Applications (NCSA) der Universität von Illinois in Urbana-Champaign entwickelt wurde. Aufgrund nachlassender Aktivitäten seitens der NCSA-Entwickler entschlossen sich 1994 der damals 21-jährige Brian Behlendorf und andere Programmierer dazu, die von der NCSA

---

<sup>65</sup> Vgl. Müller (1999), S. 8.

<sup>66</sup> Vgl. Nüttgens/Tesei (2001a), S. 3. In zahlreichen Beiträgen und Veröffentlichungen konnte sich der Begriff *Open Source* innerhalb nur weniger Monate gegen *freie Software* durchsetzen.

<sup>67</sup> Vgl. Abschnitt 2.1.1.

<sup>68</sup> Vgl. Perens (2001).

<sup>69</sup> Vgl. Müller (1999), S. 12 und Lakhani/von Hippel (2001), S. 7.

<sup>70</sup> Durch die Möglichkeit, GNU/Linux aus dem Internet zu beziehen und entsprechende Datenträger beliebig oft zu kopieren, sind Schätzungen über die Anzahl der Installationen nahezu unmöglich. Prognosen für das Jahr 2000 reichen von 9 Mio. bis zu 20 Mio. GNU/Linux-Installationen weltweit. Vgl. Flynn (2001) und Grassmuck (2001b).

<sup>71</sup> Aufgrund der folgenden Ausführungen ist anzunehmen, dass nahezu jeder Anwender eines beliebigen Internetdienstes direkt oder indirekt freie Software nutzt.

<sup>72</sup> Vgl. The Apache Software Foundation (Hrsg.) (2001b).

<sup>73</sup> Ein *Daemon* (Disk And Execution MONitor) ist ein Programm, das permanent im Hintergrund eines Computersystems läuft und darauf wartet, bei bestimmten Systemereignissen aktiv zu werden, um anschließend eine zuvor definierte Aufgabe auszuführen. Vgl. Tanenbaum (1987), S. 121.

mitunter unberücksichtigten Korrekturen und Erweiterungen zahlreicher Anwender zu sammeln und selbst in den frei zugänglichen Quellcode des `httpd` zu integrieren<sup>74</sup>. Ergebnis dieser Bemühungen ist der Apache HTTP-Server, dessen erstes Release 1995 veröffentlicht wurde. Innerhalb eines Jahres wurde der `httpd` vom NCSA von Platz eins der Liste der am häufigsten installierten HTTP-Server verdrängt<sup>75</sup>. Gegenwärtig gilt der Apache als der am weitesten verbreitete Webserver weltweit<sup>76</sup>.

### 2.3.2 Sendmail

Eric Allman entwickelte 1981 an der Universität von Kalifornien in Berkeley mit Sendmail den noch immer dominierenden Message Transport Agent (MTA)<sup>77</sup> im Internet. Nach einer Studie von Daniel J. Bernstein an der University of Chicago aus dem Jahr 2000 läuft Sendmail auf 47% aller Internet Mail-Server und leitet somit die meisten der über das Internet versendeten Nachrichten weiter<sup>78</sup>. 1998 gründeten Allman und Greg Olson mit der Unterstützung privater Investoren das gleichnamige Unternehmen Sendmail, um neben der freien eine erweiterte kommerzielle Version der Software anzubieten und durch deren Verkaufserlöse den Fortbestand und die Weiterentwicklung von Sendmail zu sichern<sup>79</sup>.

### 2.3.3 Perl

1987 veröffentlichte Larry Wall die erste Version von Perl (Practical Extraction and Report Language). Ziel von Wall war es, mit Perl eine Skriptsprache zu entwickeln, die es ermöglichen sollte, beliebige Textdateien einzulesen, Informationen aus diesen Textdateien zu extrahieren und einen Bericht basierend auf diesen Informationen zu generieren<sup>80</sup>. Mit den Jahren arbeiteten zahlreiche Programmierer an der Weiterentwicklung von Perl und erhöhten deren Leistungsumfang beträchtlich<sup>81</sup>. Das CPAN (Comprehensive Perl Archive Network), ein Onlineverzeichnis für Perl-Quellcode, Dokumentationen, Skripte sowie Module und Erweiterungen, gilt als beispielhaft für die Kooperation innerhalb der Open Source-Community. Hauptsächlich wird Perl als Skriptsprache für die CGI-Programmierung zur dynamischen Gestaltung von HTML-

<sup>74</sup> Vgl. Behlendorf (1999), S. 158 und Lerner/Tirole (2000), S. 10 f. Aufgrund der Vielzahl der Korrekturen und Erweiterungen (Patches) erhielt das Projekt den Namen *A Patchy Server*, aus dem schließlich *Apache* wurde.

<sup>75</sup> Vgl. O'Reilly & Associates (Hrsg.) (1999), S. 26 f.

<sup>76</sup> Dem *Netcraft Web Server Survey* vom August 2001 zufolge entfallen auf den Apache HTTP-Server ca. 60% der weltweit installierten Webserver. Vgl. Netcraft (Hrsg.) (2001).

<sup>77</sup> Ein MTA ist für die Verteilung von Nachrichten innerhalb eines Netzwerks zuständig.

<sup>78</sup> Vgl. Bernstein (2001).

<sup>79</sup> Vgl. O'Reilly & Associates (Hrsg.) (1999), S. 25 f.

<sup>80</sup> Vgl. Duncan/Hull (2001).

<sup>81</sup> Die aktuelle Version Perl 5.x verfügt bspw. über objektorientierte Features, vgl. Perl Mongers (Hrsg.) (2001).



Seiten<sup>82</sup> und für Aufgaben der Systemadministration eingesetzt. Schätzungen gehen von mehr als einer Million Perl-Anwendern weltweit aus<sup>83</sup>.

---

<sup>82</sup> Große Websites wie Yahoo, Amazon und Netscape setzen Perl ein, um interaktive Dienste zur Verfügung zu stellen. Perl ist die am häufigsten benutzte Programmiersprache zur Entwicklung von Internetdiensten und interaktiven Datenbankabfragen. Vgl. O'Reilly & Associates (Hrsg.) (1999), S. 28.

<sup>83</sup> Vgl. Nüttgens/Tesei (2001a), S. 16.

# Kapitel 3

## Open Source Software-Projekte

Bevor auf die Organisation, die DV-technische Unterstützung und den typischen Verlauf eines Open Source Software-Projekts näher eingegangen wird, sollen zunächst einige grundlegende Aspekte derartiger Projekte vorgestellt werden.

Hierzu zählen ihre spezifischen Eigenschaften, die hinsichtlich der zunehmenden Verbreitung freier Software nicht unwesentliche Frage nach den Gründen für eine Teilnahme an einem solchen Projekt, Voraussetzungen für erfolgreiche OSS-Projekte, Aspekte der Qualitätssicherung sowie Vorgehensweisen zur Lösung von Entscheidungsproblemen innerhalb einer projektspezifischen Community.

### 3.1 Übergreifende Aspekte

#### 3.1.1 Spezifische Eigenschaften

Trotz der nahezu unüberschaubaren Vielzahl freier Softwareprojekte und der Tatsache, dass es innerhalb der Open Source-Community so gut wie keine Reglementierungen gibt, existieren einige für derartige Softwareprojekte charakteristische Merkmale.

Diese Eigenschaften unterscheiden sich teilweise wesentlich von den für Unternehmen der Softwareindustrie üblichen Charakteristika und sollen aus diesem Grund nachfolgend kurz dargestellt werden.

##### 3.1.1.1 Projektbeginn aufgrund individueller Bedürfnisse

Der Beginn eines freien Softwareprojekts basiert häufig auf dem Vorhaben, eine vorliegende Problemstellung – wie bspw. die Automatisierung alltäglicher Vorgänge – zu lösen. Da bereits existierende Software mitunter nicht den gestellten Anforderungen genügt, beginnen viele Programmierer damit, eigene Lösungen zu entwickeln in der Annahme, dass es weitere Anwender gibt, die sich für diese Problemstellung interessieren und bei der Lösung helfen können<sup>1</sup>. Freie Software entsteht nicht auf Anweisung

---

<sup>1</sup> Vgl. Grassmuck (2001a).

eines Vorgesetzten, sondern vielmehr als Lösung für die persönlichen und alltäglichen Probleme des Initiators und verbreitet sich, weil sich das Problem als typisch für eine Vielzahl von Benutzern herausstellt<sup>2</sup>.

#### 3.1.1.2 Weltweite Verteilung der Teilnehmer

In Open Source-Communities arbeiten oftmals Menschen, die sich niemals persönlich getroffen haben und lediglich über das Internet miteinander kommunizieren, an einem Projekt mit<sup>3</sup>. Die Teilnahme an einem interessierenden Projekt ist durch das Internet nicht auf Gebäude oder Länder beschränkt, Menschen aus der ganzen Welt können sich daran beteiligen. I.d.R. ist es hierfür ausreichend, sich in eine entsprechende Mailingliste<sup>4</sup> einzutragen.

#### 3.1.1.3 Unentgeltliche und offene Teilnahme

Die Teilnahme an einem freien Softwareprojekt erfolgt nahezu ausnahmslos ohne finanzielle Anreize<sup>5</sup>. Teilnehmer eines derartigen Projekts können grundsätzlich jederzeit den Bereich ihrer Beteiligung selbst bestimmen<sup>6</sup>. Obwohl Unternehmen – wie z.B. die SuSE Linux AG – vereinzelt OSS-Entwickler einstellen und ihnen die Möglichkeit bieten, sich in Vollzeit um ihre Projekte zu kümmern<sup>7</sup>, bestimmt die Freizeit der Teilnehmer häufig das Ausmaß ihrer Beteiligung.

### 3.1.2 Motivationen von Teilnehmern

Die Motivationen von Menschen, sich in einem freien Softwareprojekt zu engagieren, sind individuell verschieden. Wegen der unentgeltlichen Teilnahme an einem solchen Projekt sind finanzielle Interessen jedoch auszuschließen. Aufgrund der Verschiedenartigkeit der Motivationen und der Tatsache, dass in der Open Source-Community oftmals ideologische Beweggründe vorliegen, sind hierüber keine allgemein gültigen Aussagen ableitbar<sup>8</sup>.

---

<sup>2</sup> Vgl. Lerner/Tirole (2000), S. 3. Eric S. Raymond formuliert es folgendermaßen: “Every good work of software starts by scratching a developer’s personal itch.” Raymond (2001d), S. 23.

<sup>3</sup> 1999 waren es z.B. bei Perl etwa 100 Beteiligte weltweit, bei Debian GNU/Linux ca. 500 und bei XFree86 ca. 600, vgl. O’Reilly & Associates (Hrsg.) (1999), Ronneburg (2001) sowie Hohndel (2001). Daneben gibt es unzählige Projekte, an denen – aufgrund ihrer Bedeutung, ihres Bekanntheitsgrades oder ihres Fortschritts – nur wenige, mitunter sogar nur ein einziger Entwickler auf lokaler Ebene, z.B. an einer Universität, arbeiten.

<sup>4</sup> Vgl. Abschnitt 3.3.1.

<sup>5</sup> Dirk Hohndel, Mitglied des Board of Directors’ des XFree86-Projekts, erläutert hierzu: “Wir haben keine bezahlten Mitarbeiter ... [und] basieren vollkommen auf freier Mitarbeit.” Hohndel (2001).

<sup>6</sup> Vgl. Dalheimer (2001).

<sup>7</sup> Vgl. Abschnitt 4.3.

<sup>8</sup> Theoretische Überlegungen über die Verhaltensmotive von Menschen finden sich z.B. in der Maslow’schen Motivationstheorie, vgl. Maslow (1954) und Maslow (1971).

Motive von Projektteilnehmern – insb. von Programmierern – die in diesem Zusammenhang vereinzelt genannt werden, lassen sich jedoch überblicksartig unter den folgenden Punkten zusammenfassen<sup>9</sup>.

### 3.1.2.1 Herausforderung und Begeisterung

Das Vorhaben, ein vorliegendes Problem zu lösen, ist nicht nur häufig der Beginn eines freien Softwareprojekts, sondern oftmals der Grund für die Teilnahme an einem bereits existierenden Projekt, welches die Lösung einer vergleichbaren Problemstellung verfolgt. Die Begeisterung, ihrem Hobby *Programmieren* gemeinsam mit anderen Menschen nachgehen zu können sowie die Herausforderung, ein gegebenes, oftmals technisch anspruchsvolles Problem zu lösen, sind für Softwareentwickler mitunter Motivation genug<sup>10</sup>.

Grundlage des Fortbestands freier Software ist demnach kein Vertragsverhältnis, sondern u.a. die Befriedigung und das persönliche Vergnügen der Projektteilnehmer<sup>11</sup>.

### 3.1.2.2 Prestige und Anerkennung

Ein oft genannter Grund für die Beteiligung an einem freien Softwareprojekt ist der Wunsch, durch aktive, zum Fortschritt des Projekts beitragende Mitarbeit Anerkennung und hohes Ansehen unter anderen Programmierern zu erreichen<sup>12</sup>.

In diesem Zusammenhang ist anzunehmen, dass sich Prestige und Anerkennung umso leichter erwerben lassen, je größer das Interesse der Öffentlichkeit und je höher der technische Anspruch eines Projekts ist<sup>13</sup>. Der Wert bzw. der Nutzen der von den

---

<sup>9</sup> Ähnliche Überlegungen hinsichtlich der Motivation von Softwareentwicklern finden sich in Lerner/Tirole (2000), S. 14 - 19.

<sup>10</sup> "It may well turn out that one of the most important effects of open source's success will be to teach us that play is the most economically efficient mode of creative work." Raymond (2001d), S. 61. Lakhani/von Hippel sprechen in diesem Zusammenhang von "intrinsically-rewarding tasks", Lakhani/von Hippel (2001), S. 10.

<sup>11</sup> Vgl. Glasl (2001). "In such situations, having a taillight to follow is a proxy for having strong central leadership." Valloppillil (2001).

<sup>12</sup> Matthias Dalheimer, Mitglied des K Desktop Environment-Projekts (KDE), erläutert hierzu: "Wir haben keinen Entwickler, dessen einzige Motivation das Gehalt oder der Wunsch ist, seinen Job nicht zu verlieren. ... Unsere Motivation ist der Ruhm, den man sich mit dieser Arbeit erwerben kann." Eine Konsequenz hieraus sei zudem qualitativ hochwertige Software, denn Anerkennung und hohes Ansehen seien nur erreichbar, wenn gute Arbeit geleistet wird. Vgl. Glasl (2001). Eric S. Raymond prägte für eine solche Gemeinschaft, dessen Mitglieder durch freiwillige Beiträge zu öffentlichen Projekten um ihren Status wetteifern, den Begriff *Gift Culture*. Er schreibt: "... participants compete for prestige by giving time, energy, and creativity away. ... In gift cultures, social status is determined not by what you control but by what you give away." Raymond (2001c), S. 65 und S. 80.

<sup>13</sup> Dies mag eine Erklärung dafür sein, dass diejenigen Tätigkeiten, welche eine vermeintlich geringe Herausforderung darstellen, z.B. die Erstellung einer Dokumentation, innerhalb der Open Source-Community als weniger attraktiv gelten. Für ein vergleichsweise unpopuläres und/oder wenig anspruchsvolles Projekt besteht somit die Gefahr, dass es schon bald aufgrund mangelnden Interesses eingestellt wird. Vgl. Lerner/Tirole (2000), S. 19 und S. 34 sowie Behlendorf (1999), S. 159.

einzelnen Beteiligten in ein Projekt investierten Zeit und Anstrengungen ist jedoch nur schwer messbar und damit kaum miteinander vergleichbar. Aus diesem Grund ist das Ausmaß der Anerkennung wesentlich vom Urteil der Community abhängig<sup>14</sup>.

### 3.1.2.3 Selbstverwirklichung und Lernen

Manche OSS-Entwickler sind in erster Linie daran interessiert, ihre Vorstellungen von einem Leben in freier Selbstbestimmung und von einem gemeinschaftlichen Miteinander jenseits hierarchischer Ordnungen in ihrer Arbeit umzusetzen<sup>15</sup>.

Zudem spornt nicht wenige Programmierer die Zusammenarbeit mit vielen erfahrenen Softwareentwicklern aus aller Welt an. Hierdurch kann sich ihnen die Möglichkeit bieten, ihre Programmierkenntnisse mit einem vergleichsweise geringen Aufwand durch den ständigen Austausch des Quellcodes zu verbessern<sup>16</sup>, die Aufmerksamkeit möglicher Arbeitgeber zu gewinnen oder sogar ein eigenes Unternehmen zu gründen<sup>17</sup>.

### 3.1.2.4 Persönliche Überzeugung und Idealismus

Viele Anhänger freier Software sind darum bemüht, ihre Vision von einer Welt, in der Software nicht aufgrund der Entscheidungen einiger weniger Unternehmen, sondern uneingeschränkt von jedem und zu jeder Zeit genutzt und verteilt werden darf, umzusetzen und zu verbreiten<sup>18</sup>. Richard M. Stallmans Überzeugung, die ihn dazu veranlasste, das GNU-Projekt zu gründen, war u.a. davon geprägt, das kooperative Wesen der damaligen Software-Sharing Communities<sup>19</sup> – eine offene Zusammenarbeit und den freien Austausch von Informationen zum Wohle der Gemeinschaft – zu bewahren.

Schließlich fühlen sich OSS-Programmierer vereinzelt dazu verpflichtet, durch das Einbringen ihrer Kenntnisse und Fähigkeiten in Form von Beiträgen für entsprechende Projekte der Gemeinschaft einen Teil der Leistungen zurückzugeben, die sie selbst

<sup>14</sup> Vgl. Raymond (2001c), S. 84 f. Innerhalb der Open Source-Community gilt es daher als schwerwiegendes Fehlverhalten, jemanden aus der Liste der Projektteilnehmer zu streichen, vgl. Lerner/Tirole (2000), S. 19 f.

<sup>15</sup> Vgl. Glasl (2001).

<sup>16</sup> In ihrer Untersuchung über die Apache Newsgroup *comp.infosystems.www.servers.unix* (CIWS-U) kommen Karim Lakhani und Eric von Hippel zu folgendem Ergebnis: "...information providers (and seekers) report that the most important reason they read CIWS-U is to learn: they gain valuable information from reading about problems other users are encountering, and how these might be solved." Lakhani/von Hippel (2001), S. 23.

<sup>17</sup> Sun Microsystems und Netscape Communications wurden bspw. durch die ehemaligen OSS-Entwickler William Joy bzw. Jim Clark gegründet, vgl. Lerner/Tirole (2000), S. 14 f. und Lerner/Tirole (2001), S. 822 f.

<sup>18</sup> Matthias Ettrich, Begründer des KDE-Projekts, führt über einen solchen Verfechter freier Software aus: "Das hehre Ziel, mehr Freiheit in die Welt zu bringen, wiegt ihm mehr als sein persönlicher Vorteil." Ettrich (2001).

<sup>19</sup> Vgl. Abschnitt 2.2.

von ihr bekommen haben<sup>20</sup>.

Volker Grassmuck fasst die Motivation vieler Open Source Software-Entwickler folgendermaßen zusammen: “Statt also aus dem Besitz und seiner kommerziellen Verwendung einen Vorteil zu ziehen, übergeben die Programmiererinnen und Programmierer freier Software ihr Werk der Öffentlichkeit, auf dass es bewundert, kritisiert, benutzt und weiterentwickelt werde.” Das Ergebnis “ist eine Schöpfung, die man mit anderen teilen kann<sup>21</sup>.”

Voraussetzung für die Aufrechterhaltung der Bereitschaft, ohne finanzielle Anreize in einem freien Softwareprojekt mitzuarbeiten, ist ein Schutzmechanismus, welcher verhindert, dass die gemeinsame Arbeit von vielen Freiwilligen durch Dritte ausgenutzt werden kann.

Die Geschichte von UNIX ist ein Beispiel hierfür. Solange UNIX weitgehend freigelegt und verändert werden durfte, fanden sich viele Interessierte, die zu dessen Weiterentwicklung beitrugen. Nach der Kommerzialisierung von UNIX jedoch brachten Unternehmen eine Vielzahl proprietärer Varianten hervor, welche auf dem ehemals *freien* und gemeinschaftlich weiterentwickelten UNIX basierten.

Das damit verbundene, sich verlierende Interesse an UNIX seitens der freien Entwicklergemeinschaft lag nicht nur an den hohen Lizenzgebühren und den restriktiven Nutzungsbedingungen, sondern gleichermaßen an der Enttäuschung über die Privatisierung und Kommerzialisierung einer Software, an der zahllose Menschen in der ganzen Welt mitgearbeitet hatten<sup>22</sup>.

Die Open Source-Community sollte schon aus Eigeninteresse versuchen zu verhindern, dass Personen oder Unternehmen ihre eigenen, auf freier Software basierenden Weiterentwicklungen nicht mit der Gemeinschaft teilen und somit die *Ressource* Motivation gefährden<sup>23</sup>. Einen solchen Schutz kann z.B. die GPL bieten.

### 3.1.3 Voraussetzungen

Neben der zwingenden Notwendigkeit, den Quellcode der Software einsehen und modifizieren zu können sowie der Verfügbarkeit leistungsfähiger Kommunikationsmittel, wie sie bspw. das Internet bietet, lassen sich folgende Voraussetzungen für eine erfolgreiche, verteilte Softwareentwicklung innerhalb der Open Source-Community beobachten.

---

<sup>20</sup> Vgl. Lerner/Tirole (2001), S. 822. Rishab Aiyer Ghosh begründet eine solche Überzeugung von OSS-Entwicklern damit, dass sie ihre Projektbeiträge als eine Art Bezahlung für die *Ideen* verstehen, die sie von der Community erhalten haben. Jeder Teilnehmer erziele somit einen mehr als fairen Ertrag in Form der kombinierten Beiträge anderer. Vgl. Ghosh (2001).

<sup>21</sup> Grassmuck (2001a).

<sup>22</sup> Vgl. Raymond (2001b), S. 10 - 15.

<sup>23</sup> Vgl. Grassmuck (2001c).

### 3.1.3.1 Veröffentlichung einer bereits lauffähigen Version

Der Open Source-Community sollte ein lauffähiges und testbares Programm vorgelegt werden, um daran ansetzend Verbesserungen und Weiterentwicklungen vornehmen zu können. Die zu veröffentlichende Software muss nicht notwendigerweise besonders stabil, fehlerfrei oder gut dokumentiert sein, sie sollte jedoch bereits funktionstüchtig sein und einige wichtige Teile desjenigen Problems lösen, für das sie ursprünglich entwickelt wurde<sup>24</sup>.

Darüber hinaus kann es in diesem Zusammenhang für ein freies Softwareprojekt entscheidend sein, in seiner anfänglichen Entwicklungsphase einen kontinuierlichen Fortschritt erkennen zu lassen. Bleibt dieser Fortschritt aus besteht die Möglichkeit, dass Entwickler ihr Interesse verlieren und sich einem anderen Projekt zuwenden<sup>25</sup>.

### 3.1.3.2 Überlegtes Verhalten des Projektbetreuers

Um eine Entwicklergemeinde aufzubauen muss man potenzielle Teilnehmer begeistern und für sein Projekt interessieren können. Der persönliche Eindruck und das Verhalten des Projektbetreuers<sup>26</sup> sind nicht zuletzt ausschlaggebend dafür, möglichst viele Anwender für das Projekt zu gewinnen und diese so zu motivieren, dass sie ihre Kenntnisse und Fähigkeiten in das Projekt investieren<sup>27</sup>. Für einen Maintainer bedeutet dies einerseits, dass er vereinzelt auch solche Vorschläge und Ergänzungen berücksichtigen sollte, die sich nicht gänzlich mit seinen Vorstellungen decken, und dass er andererseits seine Entscheidungen jederzeit sachlich und detailliert darlegt<sup>28</sup>.

Abgesehen von der nötigen Kompetenz sollte der Betreuer eines OSS-Projekts demnach über die Fähigkeit verfügen, mit Menschen umgehen und kommunizieren zu können, um eine natürliche Autorität jenseits vertraglicher Regelungen aufzubauen.

### 3.1.3.3 Umfassende DV-technische Unterstützung

Bei einem offenen Softwareprojekt, welches hauptsächlich auf der Unterstützung Freiwilliger basiert, können zweckmäßige und komfortable Möglichkeiten der Beteiligung

<sup>24</sup> Vgl. Fogel (2000), S. 129, Lerner/Tirole (2000), S. 19 und S. 21 sowie The K Desktop Environment (Hrsg.) (2001a). Die Funktionstüchtigkeit der ersten Versionen kann mitunter von entscheidender Bedeutung für den Erfolg des Projekts sein. Jamie Zawinski, ehemaliger Angestellter von Netscape Communications, behauptet, einer der wesentlichen Gründe für das vermeintliche Scheitern des Mozilla-Projekts sei, dass der ursprünglich herausgegebene Quellcode aufgrund zu vieler Fehler nahezu unbrauchbar war. Vgl. Zawinski (2001).

<sup>25</sup> Vgl. Fogel (2000), S. 126 und Köppen/Nüttgens (2000), S. 234.

<sup>26</sup> Der Initiator bzw. Betreuer eines (Teil-)Projekts ist in der OSS-Terminologie dessen *Maintainer*.

<sup>27</sup> "If you treat your beta-testers as if they're your most valuable resource, they will respond by becoming your most valuable resource." Raymond (2001d), S. 38. Vgl. ebenso Fogel (2000), S. 139 ff.

<sup>28</sup> Vgl. Lerner/Tirole (2000), S. 23 f.

oftmals entscheidend dafür sein, ob ein Entwickler aus der Community etwas zu diesem Projekt beitragen wird oder nicht<sup>29</sup>.

Es ist demnach anzunehmen, dass eine ansprechende und umfassende DV-technische Unterstützung<sup>30</sup> ein wesentliches Mittel für eine erfolgreiche Abgrenzung eines OSS-Projekts gegenüber vergleichbaren, um Unterstützung aus der Community konkurrierenden Projekten darstellen kann.

#### 3.1.3.4 Modulare Projektstruktur

Freie Softwareprojekte besitzen häufig eine modulare Struktur, welche es erlaubt, die Funktionalität der Software zu erweitern ohne Änderungen an den bereits existierenden Basisfunktionen durchführen zu müssen. Ein Maintainer kann sich somit die Kontrolle über das Kernprojekt erhalten und eine koordinierende Verantwortung darüber übernehmen – wie es bspw. Linus Torvalds mit dem Linux-Kernel praktiziert, indem er die endgültige Entscheidung über die Integration neuer Funktionalitäten trifft<sup>31</sup>.

Die Tatsache, dass viele OSS-Projekte eine derartige modulare Struktur aufweisen, beruht nicht zuletzt auf einer Entscheidung der UNIX-Entwickler. Anstelle eines komplexen, monolithischen Systems sollte UNIX aus kleinen, einfachen und effizienten Werkzeugen bestehen, welche miteinander kombiniert werden können, um komplexere Aufgaben zu lösen. Diese Strukturentscheidung wurde von freien Softwareprojekten weitgehend übernommen. Verschiedene Module können somit durch kleine, unabhängige Gruppen parallel entwickelt werden, wodurch der Aufwand an Koordination und Kommunikation begrenzt und der Projektfortschritt beschleunigt werden kann<sup>32</sup>.

#### 3.1.4 Qualitätsmanagement

Zur Beurteilung der Qualität einer Software bzw. der Güte des Software-Entwicklungsprozesses existieren unterschiedliche Kriterien. In Tabelle 3.1.4/1 wird eine Abgrenzung verschiedener Merkmale nach dem Gegenstand ihrer primären Anwendbarkeit – entweder auf das Produkt *Software* oder auf den Prozess ihrer Erstellung – vorgenommen<sup>33</sup>. Eine solche Abgrenzung mag häufig nur schwer für ein konkretes Projekt aufrechtzuerhalten sein, jedoch trägt sie dem Umstand Rechnung, dass sich das Qualitätsmanagement sowohl unter einem technischen als auch unter einem verhaltensorientierten Aspekt betrachten lässt. Während der erstgenann-

---

<sup>29</sup> Vgl. Fogel (2000), S. 33.

<sup>30</sup> Vgl. Abschnitt 3.3.

<sup>31</sup> Vgl. Grassmuck (2001a) und Nüttgens/Tesei (2001b), S. 6. Eine solche modulare Struktur lässt sich ebenfalls sehr gut beim Apache-Projekt beobachten, vgl. Behlendorf (1999), S. 150 f.

<sup>32</sup> Vgl. Grassmuck (2001a) und Lerner/Tirole (2000), S. 32.

<sup>33</sup> Vgl. Ghezzi/Jazayeri/Mandrioli (1991), S. 18 - 36. Ähnliche softwarebezogene Qualitätsmerkmale finden sich in Balzert (2001), S. 1102 f. und in Schwander (1996), S. 18.



Produkt	Prozess
Korrektheit, Zuverlässigkeit	Produktivität
Effizienz	Rechtzeitigkeit, Pünktlichkeit
Benutzerfreundlichkeit	Transparenz
Nachprüfbarkeit	
Erweiterbarkeit	
Wiederverwendbarkeit	
Portabilität	
Verständlichkeit	
Kompatibilität, Integrierbarkeit	

Tab. 3.1.4/1: Softwarebezogene Qualitätsmerkmale

te technisch-organisatorische (und wirtschaftliche) Möglichkeiten zur Gewährleistung der Softwarequalität betrifft, hat der letztgenannte Aspekt das Verhalten der Projektbeteiligten während des Entwicklungsprozesses zum Gegenstand<sup>34</sup>. Darüber hinaus ist anzunehmen, dass der Prozess ihrer Entwicklung einen wesentlichen Einfluss auf die Qualität der Software hat<sup>35</sup>.

Hinsichtlich des Open Source Software-Modells sollen nachfolgend die verschiedenen Qualitätsmerkmale bzgl. des Produkts *Software* unter dem Aspekt der Software-Qualitätssicherung zusammengefasst und diejenigen bzgl. des Prozesses der Softwareentwicklung einzeln untersucht werden.

#### 3.1.4.1 Software-Qualitätssicherung

Das OSS-Modell unterscheidet sich in der Art und Weise der Software-Qualitätssicherung – also der Erfüllung der o.g. produktbezogenen Qualitätsmerkmale – wesentlich von der für Unternehmen der Softwareindustrie typischen Vorgehensweise. Letztere bestand in der Vergangenheit oftmals darin, vor der Veröffentlichung der nahezu fertig erstellten Software Fehlerbeschreibungen aus der Sicht des Anwenders nachzuvollziehen, die Ursachen der Fehler zu lokalisieren und die Korrekturen wiederum zu prüfen bzw. von ausgesuchten externen Testern prüfen zu lassen<sup>36</sup>. In Ergänzung zu einer solchen Qualitätskontrolle gegen Ende des Software-Entwicklungsprozesses wird die Qualitätssicherung in Softwareunternehmen zunehmend als integrierter und unternehmensweiter Prozess im Rahmen eines umfassenden Qualitätsmanagement-

<sup>34</sup> Der verhaltensorientierte Aspekt betrifft weniger eine bestimmte Geisteshaltung oder Denkweise, sondern vielmehr das konkrete Verhalten aufgrund dieser Überzeugung. Vgl. Schwander (1996), S. 140 f.

<sup>35</sup> Vgl. Yeh (1993), S. xx.

<sup>36</sup> Vgl. Lerner/Tirole (2000), S. 18 und Deutsche Gesellschaft für Qualität (Hrsg.) (1995), S. 18 - 26 und S. 87 f.

Systems verstanden<sup>37</sup>.

In der OSS-Entwicklung dagegen werden Fehler und Probleme als *triviale Phänomene*<sup>38</sup> betrachtet, die durch eine genügend große Entwicklergemeinde während des Prozesses der Softwareentwicklung schnell identifiziert und behoben werden können. Erfahrungen haben gezeigt, dass durch eine derartige “vernetzte Qualitätskontrolle”<sup>39</sup>, welche frühzeitig und entwicklungsbegleitend die Fähigkeiten und Kompetenzen vieler Anwender und Entwickler zur Qualitätssicherung nutzt, Fehlfunktionen der Software im Vergleich zu nicht-freier Software auf einem wesentlich geringeren Niveau gehalten werden können<sup>40</sup>.

Eines der stärksten Argumente für das Open Source Software-Modell ist demnach die dezentralisierte, unentwegte Überprüfung und Kritik durch andere, unabhängige Entwickler und fachkundige Anwender. Es ist anzunehmen, dass dieser *Peer Review*-Prozess<sup>41</sup> ein wesentlicher Grund für die oft erwähnte Stabilität und Qualität freier Software ist<sup>42</sup>.

### 3.1.4.2 Sicherung der Qualität des Software-Entwicklungsprozesses

#### Produktivität

Dem Open Source-Modell wird vereinzelt mangelnde Produktivität und Effizienz hinsichtlich der Reaktion auf technologische Veränderungen vorgeworfen<sup>43</sup>. In den *Halloween-Dokumenten*<sup>44</sup> z.B. urteilt Microsoft über die OSS-Entwicklung, dass sie nur Ideen nachahmen könne, die bereits allgemeiner Standard in der Softwareentwicklung sind. Aufgrund unzureichenden Managements sei das *Erreichen neuer Fronten*<sup>45</sup>,

<sup>37</sup> Vgl. Deutsche Gesellschaft für Qualität (Hrsg.) (1995), S. 18 - 26.

<sup>38</sup> Vgl. Raymond (2001d), S. 30 f.

<sup>39</sup> Köppen/Nüttgens (2000), S. 231.

<sup>40</sup> Vgl. Köppen/Nüttgens (2000), S. 231. Sozialwissenschaftler fanden schon vor Jahren heraus, dass die gemittelte Meinung einer Menge von gleich kompetenten Beobachtern zuverlässigere Vorhersagen erlaubt als die eines einzelnen, willkürlich bestimmten Beobachters. Sie prägten hierfür den Begriff *Delphi-Effekt*. Vgl. zum Delphi-Effekt bzw. zur Delphi-Methode bspw. Linstone/Turoff (Hrsg.) (1975).

<sup>41</sup> Die Open Source-Community befindet sich hiermit – wohl eher unbeabsichtigt – in der Tradition der akademischen Wissenschaftsverfassung, wie sie in der Gelehrtenrepublik des 19. Jahrhunderts entstand. In ihrem Grundsatz von der *Trennung von Erkenntnis und Eigentum* fordert sie u.a. die Veröffentlichung von Forschungsergebnissen, um sie in einem *Peer Review*-Prozess überprüfen, kritisieren und fortschreiben zu können. Vgl. Grassmuck (2001a).

<sup>42</sup> Vgl. z.B. BMWi (Hrsg.) (2001), S. 21.

<sup>43</sup> Vgl. Raymond (2001f), S. 222 f.

<sup>44</sup> Die sog. *Halloween-Dokumente* aus dem Jahr 1998 waren zunächst Microsoft-interne und vertrauliche Analysen von OSS im Allgemeinen und von GNU/Linux im Speziellen, wurden jedoch Eric S. Raymond zugespielt und anschließend veröffentlicht. Microsofts ursprünglicher Titel des ersten Teils lautet: “Open Source Software: A (New?) Development Methodology”, vgl. Valloppillil (2001). Weltweit bekannt wurde dieses Dokument vor allem aufgrund der dort beschriebenen Strategien, mit denen Microsoft einer weiteren Verbreitung von GNU/Linux entgegenwirken kann.

<sup>45</sup> Vgl. Valloppillil (2001).

also darüber hinausgehender Fortschritt, nahezu unmöglich<sup>46</sup>.

Aufgrund der Tatsache, dass Software in erster Linie auf Bedürfnissen und Ideen basiert<sup>47</sup> und das Ergebnis kreativer Tätigkeiten von i.d.R. mehreren Personen ist, liegt die Vermutung nahe, dass sich Fortschritt und Innovationen am einfachsten dann erreichen lassen wenn es gelingt, möglichst viele Menschen heranzuziehen, die zu solchen Ideen und zu kreativer Gestaltung fähig sind<sup>48</sup>. Wichtig erscheint in diesem Zusammenhang allein das Vermögen, diese Einsichten und Ideen zu fördern und umzusetzen. Hiervon ist jedoch sowohl die traditionelle Softwareentwicklung, wie sie in Unternehmen der Softwareindustrie üblich ist, als auch die OSS-Entwicklung betroffen. Die o.g. Kritik ist somit keineswegs eine Unzulänglichkeit der Open Source-Community, sondern vielmehr ein allgemeines Problem der Softwareentwicklung.

### Rechtzeitigkeit, Pünktlichkeit

Da freie Softwareprojekte nahezu ausnahmslos keinem Terminplan unterliegen<sup>49</sup>, sind Rechtzeitigkeit bzw. Pünktlichkeit für derartige Projekte nur von geringer Bedeutung.

### Transparenz

Unzureichende Transparenz und Übersichtlichkeit sowie eine damit einhergehende ineffiziente Softwareentwicklung durch eine nur lockere, weitgehend unkoordinierte Zusammenarbeit ohne feste Aufgabenverteilung ist ein weiterer, oftmals vorgetragener Kritikpunkt gegenüber dem OSS-Modell<sup>50</sup>. Hierdurch bestehe zudem eine hohe Wahrscheinlichkeit für einen sog. *Code Fork*<sup>51</sup>, welcher die Entstehung mehrerer instabiler und zueinander inkompatibler Versionen einer Software fördere<sup>52</sup>.

Dem entgegen steht jedoch die Tatsache, dass sich durch die zahlreichen Möglichkeiten des Informationsaustausches Softwareentwickler und andere Interessierte jederzeit über den aktuellen Status des Projekts informieren können<sup>53</sup>. Darüber hinaus

<sup>46</sup> Im selben Dokument jedoch widerspricht der Autor z.T. seiner Aussage, indem er darlegt, dass neuartige Ideen der Grundlagenforschung als erstes unter GNU/Linux implementiert würden und verfügbar seien, vgl. Valloppillil (2001).

<sup>47</sup> Vgl. Grassmuck (2001a) und O'Reilly (2001).

<sup>48</sup> "Insight comes from individuals." Raymond (2001f), S. 222. Vgl. zudem Schwander (1996), S. 97 und S. 127 f. Lerner/Tirole entgegnen dem: "...the probability that the innovation is made need not increase with the number of developers, as free-riding is stronger when the number of potential developers increases." Lerner/Tirole (2000), S. 18. Der Erfolg freier Software lässt jedoch vermuten, dass – neben den verhaltensorientierten Aspekten – die Rahmenbedingungen, welche die Entwicklung derartiger Software auszeichnen, wie z.B. die uneingeschränkte Modifizierbarkeit des Quellcodes und die für jeden offen stehende Teilnahme, einen positiven Einfluss auf die Wahrscheinlichkeit innovativer Entwicklungen haben können.

<sup>49</sup> Vgl. Abschnitt 3.2.2.2.

<sup>50</sup> Vgl. Fogel (2000), S. 27 und S. 207 - 211.

<sup>51</sup> Vgl. Abschnitt 3.1.5.

<sup>52</sup> Vgl. Mundie (2001).

<sup>53</sup> Benutzer können bspw. per eMail in direkten Dialog mit den Entwicklern bzw. Betreuern treten.

hat es sich innerhalb der Open Source-Community eingebürgert, bei Unklarheiten die Wünsche und Entscheidungen des Maintainers, sofern sie plausibel und entsprechend detailliert dargelegt wurden, zu respektieren<sup>54</sup>.

In der Praxis kommt diesem Vorwurf gegenüber der OSS-Entwicklung nur eine zu vernachlässigende Bedeutung zu.

Allein die Tatsache, dass sich die Art und Weise der Qualitätssicherung in einem freien Softwareprojekt z.T. wesentlich von einer herkömmlichen, systematisch geplanten Vorgehensweise unterscheidet, erlaubt jedoch noch keine Aussage darüber, ob das Open Source-Modell in dieser Hinsicht der traditionellen Softwareentwicklung grundsätzlich unter- oder überlegen ist. Für beide Ansätze scheint es vielmehr entscheidend zu sein, inwieweit es ihnen gelingt, auf effiziente Weise die zur Verfügung stehenden Mittel zu nutzen sowie die sich bietenden Möglichkeiten umzusetzen. Sowohl freie als auch kommerzielle Softwareprojekte sind demnach wesentlich von den Leistungen und Beiträgen ihrer Teilnehmer abhängig.

### 3.1.5 Konfliktmanagement

Bei einer Vielzahl zu treffender Entscheidungen stimmen die Meinungen der Mitglieder einer projektspezifischen Community nur selten überein. Aufgrund des Fehlens einer entscheidungsberechtigten Leitung werden Konflikte innerhalb eines freien Softwareprojekts i.d.R. derart gelöst, dass die Beteiligten so lange darüber diskutieren, bis ein von allen tragfähiger Kompromiss erreicht werden konnte oder bis sich eine signifikante Mehrheit für einen Vorschlag ausgesprochen hat<sup>55</sup>.

Sollte auf diese Weise dennoch keine Einigung erzielt werden können, wird entweder die endgültige Entscheidung dem Maintainer bzw. dem sogenannten *Core Team*<sup>56</sup> zugestanden oder – dem Prinzip der Seniorität folgend – zugunsten derjenigen Partei entschieden, die den bislang größten Anteil zum Fortschritt des Projekts beigetragen hat<sup>57</sup>.

Gefahr und Chance zugleich ist für ein freies Softwareprojekt dann gegeben, wenn sich die Beteiligten trotz allem nicht einigen können oder wenn es Entwickler gibt, die mit den getroffenen Entscheidungen nicht einverstanden sind. Durch den frei verfügbaren Quellcode besteht gerade in einem solchen Fall die Möglichkeit, dass

---

<sup>54</sup> Vgl. Abschnitt 3.1.5.

<sup>55</sup> Vgl. Glasl (2001) und Eilebrecht (2001). Dave Clark von der Internet Engineering Task Force (IETF) formulierte es folgendermaßen: “We reject: kings, presidents, and voting. We believe in: rough consensus and running code.” Borsook (2001). Die genannten Entscheidungsprobleme können z.B. die Integration verschiedener Funktionalitäten, die Nachfolge des aktuellen Maintainers oder die Art und Weise der Anerkennung von Projektbeiträgen betreffen.

<sup>56</sup> Vgl. Abschnitt 3.2.1.2.

<sup>57</sup> Vgl. Raymond (2001c), S. 103 f.

ein oder mehrere Programmierer ein eigenes, auf dem bisherigen basierendes Projekt beginnen. Ein solcher Vorgang wird in der OSS-Terminologie als *Code Fork* bezeichnet<sup>58</sup>.

Durch die jeweils verkleinerten projektspezifischen Communities und aufgrund der Annahme, dass die Projektfortschritte nicht deckungsgleich sind, besteht die Gefahr darin, dass sich das Projekt an sich verzögert und unter Umständen nicht von der besten Entwicklung profitieren wird.

Dem gegenüber bietet sich die Gelegenheit, dass ein Projekt, dessen Maintainer sich als überfordert erweist oder Vorschläge und Kritik seitens der übrigen Teilnehmer nicht berücksichtigt<sup>59</sup>, von anderen Entwicklern übernommen und den Anregungen entsprechend fortgeführt werden kann.

Welches der Teilprojekte sich letztendlich durchsetzen wird, entscheiden in erster Linie die Anwender. Ein solcher *Code Fork* ist somit einerseits eine der schärfsten Formen der Kritik durch die Community sowie andererseits eine faire Herausforderung für ein Projekt, seinen Betreuer und alle weiteren Beteiligten<sup>60</sup>.

## 3.2 Organisation

Unter dem Begriff *Organisation* wird die Gesamtheit derjenigen Maßnahmen verstanden, welche ein soziales System arbeitsteilig strukturieren und die Aktivitäten der zum System gehörenden Menschen ordnen<sup>61</sup>.

Die Organisationsstruktur eines Unternehmens als das Ergebnis sämtlicher organisatorischer Gestaltungshandlungen resultiert gewöhnlich aus einer planvollen und systematischen Vorgehensweise der Unternehmensleitung.

In freien Softwareprojekten dagegen gibt es keine Leitung, die über die Verteilung der Aufgaben entscheidet oder den zeitlichen Projektverlauf bestimmt, kein Vorgesetzter sagt den Teilnehmern, was zu tun ist. Insbesondere aufgrund einer häufig weltweiten Verteilung der Projektteilnehmer, ihrer in kaum einer Weise vorhersehbaren Beiträge sowie einer vereinzelt ablehnenden Haltung gegenüber hierarchischen Strukturen ist eine Organisation derartiger Projekte i.d.R. nur durch Kompromisse und Vereinbarungen möglich. Die besonderen Eigenschaften der Open Source-Community wirken sich offensichtlich nachteilig auf die Planbarkeit und Organisierbarkeit ihrer Softwareprojekte aus.

---

<sup>58</sup> Beispiele hierfür sind die Trennung des XEmacs vom GNU Emacs, diejenige des egcs vom gcc sowie die verschiedenen BSD UNIX-Varianten. Vgl. Raymond (2001c), S. 72.

<sup>59</sup> Ursachen hierfür können nicht nur in einer fachlichen Überforderung zu finden sein, sondern gleichermaßen in der Überlastung des Maintainers: "...I would like to point out that leaders of successful OSS projects get so much e-mail that just reading it may constitute a substantial workload. ... To create a successful product, you need to sacrifice time and energy." Bezroukov (2001).

<sup>60</sup> Vgl. Fogel (2000), S. 151 und Raymond (2001c), S. 85.

<sup>61</sup> Vgl. Hill/Fehlbaum/Ulrich (1994), S. 17.

Dennoch lassen sich in diesem Zusammenhang strukturierende Maßnahmen innerhalb der Open Source-Community beobachten. Diese Maßnahmen sollen im folgenden – unterteilt in die Aufbau- und die Ablauforganisation – aufgezeigt werden.

### 3.2.1 Aufbauorganisation und Konfiguration

Die Aufbauorganisation umfasst gewöhnlich die Gliederung eines Systems in Subsysteme sowie die Verteilung der (Teil-)Aufgaben auf die Systemelemente – sie betrifft somit die Bildung von Aktionseinheiten und deren Koordination<sup>62</sup>.

Neben dem einzelnen, häufig seine eigenen Interessen verfolgenden Teilnehmer als elementarer Bestandteil einer projektspezifischen Community sind in freien Softwareprojekten ab einer gewissen Komplexität weitere Subsysteme auszumachen. Diese sowie die Art und Weise der Verteilung der Aufgaben innerhalb eines solchen selbstorganisierenden Systems werden nachfolgend dargestellt.

#### 3.2.1.1 Verteilung der Aufgaben auf die Teilnehmer

Freie Softwareprojekte sind u.a. dadurch gekennzeichnet, dass es keine feste Aufgabenverteilung gibt und die Teilnehmer ihren Wirkungskreis – begünstigt durch die Aufteilung des Projekts in kleinere Einheiten (Module) – frei wählen können. Neben der Softwareentwicklung an sich können Interessierte bspw. bei der Übersetzung in andere Sprachen oder bei der Erstellung der Dokumentation helfen<sup>63</sup>.

Um jedoch die verschiedenen Bereiche eines freien Softwareprojekts gleichermaßen abzudecken, geben die unmittelbaren Projektbetreuer mitunter Anregungen, welches Teilprojekt am dringendsten Unterstützung benötigt und versuchen zudem, Teilnehmer mit einander ergänzenden Kenntnissen und Fähigkeiten zueinanderzubringen<sup>64</sup>.

#### 3.2.1.2 Bildung einer koordinierenden Projektleitung

Die Bildung einer betreuenden und koordinierenden Projektleitung – welche häufig als *Core Team* bezeichnet wird – ist oft bei größeren Projekten zu beobachten. Diese, die Initiatoren und Kernentwickler des Projekts umfassende Gruppe macht u.a. Vorschläge hinsichtlich der weiteren Entwicklung der Software, regelt die Verwaltung des Quellcodes oder kümmert sich um Fragen der Öffentlichkeitsarbeit<sup>65</sup>. Darüber hinaus bemüht sich das Core Team darum, weitere Teilnehmer für das Projekt zu

---

<sup>62</sup> Vgl. Grochla (1995), S. 24.

<sup>63</sup> Vgl. Grassmuck (2001b).

<sup>64</sup> Vgl. Dalheimer (2001).

<sup>65</sup> Vgl. Dalheimer (2001). Vereinzelt lässt sich in der Open Source-Community beobachten, dass die Betreuer eines Teilprojekts nach dem Rotationsprinzip ständig untereinander wechseln. Vgl. Lerner/Tirole (2000), S. 6 und S. 12.

gewinnen, um dessen Fortbestand zu gewährleisten und ggf. einen drohenden Code Fork abzuwenden<sup>66</sup>.

Die Aufnahme in ein bestehendes Core Team gilt für die meisten Teilnehmer eines freien Softwareprojekts als Auszeichnung, sie erfolgt jedoch i.d.R. erst aufgrund fortwährender Verdienste um die Weiterentwicklung des Projekts. Neue Mitglieder können von bisherigen Mitgliedern vorgeschlagen werden<sup>67</sup>.

### 3.2.1.3 Gründung einer institutionellen Rahmenorganisation

Bedingt durch das zunehmende Interesse der Softwareindustrie können sich für freie Softwareprojekte vereinzelt Möglichkeiten der Zusammenarbeit mit kommerziellen Unternehmen eröffnen. Um jedoch (schriftliche) Vereinbarungen mit diesen Unternehmen abschließen zu können, bedarf es i.d.R. eines institutionellen Rahmens in Form eines *offiziellen* Vertragspartners<sup>68</sup>. Bei Debian GNU/Linux ist dies z.B. die Organisation *Software in the Public Interest*<sup>69</sup>, bei Apache *The Apache Software Foundation*<sup>70</sup>. Derartige Institutionen, zu denen ebenfalls die Open Source Initiative (OSI) und die Free Software Foundation (FSF) zählen, sind keine gewinnorientierten Organisationen, sondern vielmehr – neben ihrer Eigenschaft als Vertragspartner für interessierte Unternehmen – ein möglicher Empfänger für Spenden und ähnliche Projektbeiträge. Mitunter nehmen sie sich auch rechtlichen, bspw. im Zusammenhang mit der Vertragsgestaltung stehenden Dingen an<sup>71</sup>.

## 3.2.2 Ablauforganisation und Koordination

Gegenstand der Ablauforganisation ist die Strukturierung der zu einer Aufgabenerfüllung erforderlichen Vorgänge. Dies umfasst gewöhnlich die Bildung von Arbeitsgängen, deren Zuordnung zu Aufgabenträgern und ihre zeitliche Abstimmung aufeinander<sup>72</sup>.

Bei einem freien Softwareprojekt ist jedoch – aufgrund einer verteilten, lockeren und weitgehend unkoordinierten Zusammenarbeit sowie bedingt durch die Tatsache, dass die Teilnehmer den Umfang, die Dauer und den Bereich ihrer Beteiligung

<sup>66</sup> Vgl. Lerner/Tirole (2000), S. 21.

<sup>67</sup> Vgl. Dalheimer (2001). Da die Aufnahme in das Core Team in einem solchen Fall “als Belohnung für individuelle Begabung, persönlichen Einsatz und die unbestreitbare Leistung” (Young (1961), S. 144.) erfolgt, beruhen derartige Projekte auf einer Meritokratie (*Herrschaft der Verdienten*). Vgl. Grassmuck (2001b). Plausible und überlegte Entscheidungen sind ebenfalls hinsichtlich der Zusammensetzung der Projektleitung wichtig, da sich benachteiligt fühlende Teilnehmer jederzeit ihre Beteiligung beenden können. Vgl. Bezroukov (2001).

<sup>68</sup> Die Ankündigung von IBM, mit der Apache Group zusammenarbeiten zu wollen, veranlasste letztere bspw. zu der Gründung der Apache Software Foundation. Vgl. Grassmuck (2001b).

<sup>69</sup> Vgl. *Software in the Public Interest* (Hrsg.) (2001).

<sup>70</sup> Vgl. *The Apache Software Foundation* (Hrsg.) (2001c).

<sup>71</sup> Vgl. Eilebrecht (2001).

<sup>72</sup> Vgl. Grochla (1995), S. 25.

grundsätzlich selbst bestimmen – nicht abzusehen, welcher Teilnehmer zu welchem Zeitpunkt welchen Beitrag für das Projekt leisten wird. Eine systematische Ablauforganisation ist demnach nahezu undurchführbar.

Der folgende Abschnitt konzentriert sich aus diesem Grund auf den eigentlichen Prozess der Softwareentwicklung innerhalb der Open Source-Community. Näher betrachtet werden die jeweiligen Vorgänge in den verschiedenen Phasen der Softwareentwicklung sowie diejenigen, welche die Planung und Steuerung eines freien Softwareprojekts betreffen.

### 3.2.2.1 Phasen der Softwareentwicklung

Der für kommerzielle Projekte typische Verlauf der Softwareentwicklung lässt sich in die Phasen Planung, Definition, Entwurf, Implementierung, Abnahme und Einführung sowie Wartung und Pflege einteilen<sup>73</sup>. Obwohl sich einige dieser Phasen auch in freien Softwareprojekten wiederfinden, folgt dort die Softwareentwicklung in den wenigsten Fällen einer solchen klar definierten, linearen Struktur.

Da es für freie Software ohnehin keinen Auftraggeber gibt, entfällt die Phase der Abnahme. Zudem liegen sowohl die Installation als auch die Inbetriebnahme derartiger Software i.d.R. in der Hand des Anwenders<sup>74</sup>, die Einführungsphase ist demnach nicht Gegenstand des Entwicklungsprozesses freier Software. Die Aktivitäten der Wartungs- und Pflegephase – wie z.B. Fehlerbehebung, Optimierung und Erweiterung<sup>75</sup> – schließlich sind in OSS-Projekten ohnehin ständiger Bestandteil der Softwareentwicklung.

Die noch verbleibenden Phasen lassen sich ebenfalls in freien Softwareprojekten beobachten, jedoch sind sie hier nur selten Gegenstand systematischer Planungen, wie sie in der Softwareindustrie üblich sind<sup>76</sup>. Nachfolgend sollen die für freie Softwareprojekte typischen Phasen der Softwareentwicklung vorgestellt werden.

### Projektstart

Am Anfang eines freien Softwareprojekts steht i.d.R. eine Idee – entweder eine Idee zur Lösung eines Problems bzw. einer störenden Unzulänglichkeit oder eine Vorstellung von einem nützlichen Programm, welches weitere Anwender interessieren könnte. Typischerweise wird diese Idee bereits so weit implementiert, dass ein hinreichend lauffähiges Programm vorliegt<sup>77</sup>. Dem Projektstart lassen sich demnach weitgehend die herkömmlichen Phasen der Planung und Definition sowie teilweise diejenigen des Entwurfs und der Implementierung zuordnen.

<sup>73</sup> Vgl. Balzert (2001), S. 55 und Schwander (1996), S. 1.

<sup>74</sup> Manche Unternehmen bieten jedoch kommerziellen Support für OSS an, vgl. hierzu Abschnitt 4.2.

<sup>75</sup> Vgl. Balzert (2001), S. 1090 - 1093.

<sup>76</sup> Vgl. Fogel (2000), S. 208 ff.

<sup>77</sup> Vgl. Raymond (2001d), S. 47 sowie Abschnitt 3.1.3.1.



### Ankündigung und Veröffentlichung

Diese lauffähige und vor allem testbare Software wird anschließend der Allgemeinheit zugänglich gemacht. Hierzu gehört gewöhnlich die Einrichtung einer möglichst umfassenden DV-technischen Infrastruktur, um Interessierten eine einfache und komfortable Möglichkeit der Teilnahme zu bieten. Je mehr Anwender sich schließlich für die Software oder zumindest für die dahinter stehende Idee interessieren, desto größer ist die Wahrscheinlichkeit für einen Erfolg des Projekts<sup>78</sup>.

### Entwicklung innerhalb der Open Source-Community

In seinem Essay *The Cathedral and the Bazaar* untersucht Eric S. Raymond die Vorgehensweisen und Mechanismen der OSS-Entwicklung<sup>79</sup>. Die Ergebnisse seiner Untersuchung lassen sich wie folgt zusammenfassen:

Zunächst wird die Aussicht, durch aktive Mitarbeit an einem Projekt wesentlich zu dessen Fortschritt beitragen zu können, viele Anwender, die daran interessiert sind, die Funktionsweise und den Aufbau der Software zu erforschen sowie Lösungen für gefundene Probleme zu erarbeiten, für eine Teilnahme motivieren<sup>80</sup>.

Frühzeitige und häufige Veröffentlichungen<sup>81</sup> neuer, aktualisierter Versionen der Software, in denen die Fehlerbeschreibungen und Verbesserungen der Anwender umgesetzt wurden<sup>82</sup>, können zudem dazu beitragen, dass durch einen regen Informationsaustausch die Anzahl der Anwender steigt und die Beteiligten ihr Interesse nicht verlieren. In diesem Zusammenhang kann ein Projekt schon bald eine Vielzahl von Entwicklern auf sich vereinen, wenn dessen Betreuer anspruchsvolle und herausfordernde Aufgaben der Community überlässt und nicht gänzlich selbst übernimmt<sup>83</sup>.

Durch Einbeziehung der Anwender als Mitentwickler und Ermunterung zur aktiven Mitarbeit<sup>84</sup> sowie aufgrund der Tatsache, dass unter den Anwendern viele erfahrene Programmierer sind, können schließlich Fehler schnell diagnostiziert und behoben

<sup>78</sup> Raymond erklärt hierzu: "Given enough eyeballs, all bugs are shallow." Raymond (2001d), S. 30.

<sup>79</sup> Seine Überlegungen testete er an fetchmail, einem freien Softwareprojekt, welches er selbst koordinierte und betreute. Aufgrund dieser Erfahrungen und weiteren Beobachtungen übertrug er die Ergebnisse seiner Analyse der GNU/Linux-Entwicklung auf OSS-Projekte im Allgemeinen. Raymond prägte für das OSS-Entwicklungsmodell den Begriff *Basarmethode*.

<sup>80</sup> Vgl. Lerner/Tirole (2000), S. 21 f.

<sup>81</sup> Linus Torvalds bspw. veröffentlichte anfangs mehrere aktualisierte Versionen des Linux-Kernels an nur einem Tag, vgl. Raymond (2001d), S. 29.

<sup>82</sup> OSS-Entwickler betrachten die Integration ihrer Vorschläge und Beiträge oft als *Belohnung* für ihre Mitarbeit. Vgl. Raymond (2001d), S. 30 und Behlendorf (1999), S. 161.

<sup>83</sup> Vgl. Lerner/Tirole (2000), S. 21 und Bezroukov (2001).

<sup>84</sup> Auch im kommerziellen Bereich setzt sich zunehmend die Erkenntnis durch, zukünftige Benutzer der Software frühzeitig in deren Entwicklungsprozess einzubeziehen, jedoch vorwiegend vor dem Hintergrund, die Software möglichst optimal an die Möglichkeiten und Bedürfnisse der Benutzer anzupassen sowie Anregungen für zusätzliche Funktionalitäten zu erhalten. Vgl. Schwander (1996), S. 136 ff. und Lerner/Tirole (2001), S. 822.

sowie notwendige Änderungen umgehend vorgenommen werden<sup>85</sup>.

Frederick P. Brooks, Jr.<sup>86</sup> führt in diesem Zusammenhang aus, dass Komplexität und Umfang der Kommunikation zwischen den Teilnehmern eines Softwareprojekts quadratisch mit der Anzahl der Beteiligten steigen, der Fortschritt des Projekts an sich jedoch nur linear verläuft<sup>87</sup>. Raymond entgegnet dem, dass bei einem freien Softwareprojekt nur eine unwesentliche Koordination und Kommunikation der Beteiligten untereinander nötig sei mit dem Ergebnis, dass durch eine große Anzahl interessierter und talentierter Tester und Mitentwickler *Debugging*<sup>88</sup> *parallelisierbar* sei. Die OSS-Entwicklung sei somit weitgehend unanfällig gegenüber der von Brooks formulierten Gesetzmäßigkeit und ermögliche eine schnelle und zuverlässige Softwareentwicklung<sup>89</sup>.

Die Phasen des weiteren Softwareentwurfs und der Implementierung lassen sich demnach der Entwicklung innerhalb der Open Source-Community zuordnen.

### Projektende

Ein freies Softwareprojekt endet typischerweise damit, dass das Projekt aus verlorenem Interesse, aus Zeitgründen oder weil es als abgeschlossen gilt, nicht mehr weiterverfolgt wird und sich niemand findet, der es aus beliebigen Gründen von den ursprünglichen Betreuern übernimmt und fortführt<sup>90</sup>.

#### 3.2.2.2 Projektplanung

Gegenstand der Projektplanung sind – neben den bereits in Abschnitt 3.2.1 betrachteten aufbauorganisatorischen Aspekten – im wesentlichen die Planung der Termine, der Qualität, der Ressourcen und der Kosten<sup>91</sup>. Im folgenden Abschnitt soll skizziert werden, ob und inwiefern derartige Aktivitäten für freie Softwareprojekte relevant sind.

### Terminplanung

Üblicherweise setzen sich freie Softwareprojekte keinen Zeitplan für eine Veröffent-

---

<sup>85</sup> Dementsprechend lässt sich beobachten, dass viele Anwender Korrekturen selbst vornehmen und testen, noch bevor sie diese an den Betreuer des Projekts weiterleiten. Vgl. Raymond (2001d), S. 38.

<sup>86</sup> Brooks war Manager bei IBM und Leiter des OS/360-Projekts.

<sup>87</sup> Ein Resultat seiner Untersuchungen ist als *Brooks' Gesetz* bekannt: "Adding manpower to a late software project makes it later." Brooks (1975), S. 25.

<sup>88</sup> In der DV-Terminologie bezeichnet ein *Bug* einen technischen, syntaktischen oder logischen Fehler in der Hard- oder Software. Das Lokalisieren, Analysieren und Korrigieren dieser Fehler wird *Debugging* genannt, vgl. Deutsche Gesellschaft für Qualität (Hrsg.) (1995), S. 88.

<sup>89</sup> Vgl. Raymond (2001d), S. 32 f.

<sup>90</sup> Vgl. Fogel (2000), S. 152 f.

<sup>91</sup> Vgl. Biethahn (1996), S. 195 und Mertens/Bodendorf/König/Picot/Schumann (1996), S. 196 f.

lichung neuer, *offizieller* Versionen<sup>92</sup>. Erreicht ein Projekt einen vereinbarten Status, erfolgt i.d.R. ein sog. *Code Freeze*, ab dem ausschließlich Änderungen zugelassen werden, die absolut nötig sind, um Fehler zu beheben, und keinesfalls solche, die sich destabilisierend auf die Software auswirken könnten<sup>93</sup>. Sobald nach ausgiebigen Tests alle offensichtlichen bzw. gemeldeten Fehler behoben wurden, wird das Projekt schließlich mit einer neuen Versionsnummer offiziell freigegeben, eine systematische Terminplanung liegt jedoch nicht vor.

### Qualitätsplanung

Viele OSS-Projekte präsentieren sich und ihren Fortschritt auf einer entsprechenden Website. Häufig finden sich dort eine öffentliche Fehlerdatenbank, in die Anwender einen sog. *Bug Report* mit der Fehlerbeschreibung einstellen können sowie eine öffentliche *ToDo-Liste*, die einen Überblick über zukünftige Entwicklungen und notwendige Erweiterungen liefert. Anwender können hier jederzeit Vorschläge für Ergänzungen (*Feature Request*) einbringen<sup>94</sup>.

Aspekte, die im weitesten Sinne einer Qualitätsplanung zuzuordnen sind, beschränken sich bei freien Softwareprojekten gewöhnlich auf diese beiden Mittel.

### Ressourcen- und Kostenplanung

Jeder Teilnehmer eines freien Softwareprojekts trägt die in diesem Zusammenhang anfallenden Kosten – z.B. für die Nutzung einer Internetverbindung – selbst. Kosten für eine entsprechende DV-technische Infrastruktur<sup>95</sup> werden entweder von den Projektbetreibern getragen oder von Sponsoren übernommen<sup>96</sup>. In ähnlicher Weise verhält es sich mit den Ressourcen, deren Beschaffung und Einsatz letztendlich dem einzelnen Projektteilnehmer obliegt.

Ressourcen und Kosten sind in freien Softwareprojekten somit in den wenigsten Fällen Gegenstand systematischer Planungen.

<sup>92</sup> Eine Ausnahme bildet das KDE-Projekt. Matthias Dalheimer erläutert hierzu: “Es gibt einen Rahmenplan, der den Tag der Veröffentlichung und die einzelnen Arbeitsfortschritte bestimmt.” Jedoch sei es ohne weiteres möglich, “den Tag der Veröffentlichung um einen Monat zu verschieben, wenn wir merken, dass wir sonst die angestrebte Qualität nicht erreichen können.” Vgl. Glasl (2001). Darüber hinaus ist bei einem OSS-Projekt, welches das Concurrent Versions System (CVS, vgl. Abschnitt 3.3.2.) einsetzt, ohnehin der Zugriff auf den tagesaktuellen Stand der Entwicklung möglich.

<sup>93</sup> Im Unterschied zu einem *Code Freeze* sind bei einem sog. *Feature Freeze* “kleinere Verbesserungen dann erlaubt, wenn sie an isolierter Stelle eingebaut werden und (theoretisch) den Quelltext nicht destabilisieren können.” Hierbei kann es sich z.B. um die Ausgabe von Fehlermeldungen handeln. Da sich jedoch ein *Code Freeze* nur schwer von einem *Feature Freeze* unterscheiden lässt, werden die beiden Begriffe meist synonym zueinander verwendet, vgl. Fogel (2000), S. 293.

<sup>94</sup> Vgl. bspw. The K Desktop Environment (Hrsg.) (2001b).

<sup>95</sup> Hierunter fallen bspw. Ausgaben für Hardware, etwa für leistungsfähige Rechner, oder für Leistungen, die im Zusammenhang mit der Anbindung an das Internet in Anspruch genommen werden.

<sup>96</sup> Vgl. bspw. VA Linux Systems (Hrsg.) (2001a).

Eine Projektplanung im Sinne eines systematischen, vorwegnehmenden Durchdenkens zukünftiger Aufgaben und Maßnahmen kann demnach bei einem freien Softwareprojekt nur ansatzweise festgestellt werden.

### 3.2.2.3 Projektsteuerung

Zu den wesentlichen Aufgaben der Projektsteuerung gehören einerseits die Anleitung, Motivierung und Koordination der Projektteilnehmer sowie andererseits Maßnahmen zur Durchsetzung der während der Projektplanung getroffenen Vorgaben<sup>97</sup>.

Aufgrund des Fehlens einer entsprechenden Projektleitung lassen sich führende, koordinierende und überwachende Maßnahmen allenfalls dem Maintainer bzw. dem Core Team zuschreiben. Letztendlich werden im Bedarfsfall Entscheidungen und Maßnahmen gemeinschaftlich von der projektspezifischen Community diskutiert und durchgeführt<sup>98</sup>.

## 3.3 DV-technische Unterstützung

Die spezifischen Eigenschaften der Open Source-Community beeinflussen nicht nur die Organisation ihrer Softwareprojekte. Gleichmaßen lässt sich deren DV-technische Unterstützung als Reflektion ihrer organisatorischen Merkmale auf die genannten Eigenschaften zurückführen.

Abgesehen von der eigentlichen Softwareentwicklung wird in einem freien Softwareprojekt nahezu jeder Prozess – beginnend bei der erstmaligen Ankündigung über die Bekundung der Teilnahme bis zur Veröffentlichung einzelner Versionen – durch DV-technische Mittel unterstützt. Bevor auf das Concurrent Versions System als das zentrale Werkzeug einer verteilten Softwareentwicklung eingegangen wird, sollen zunächst die für freie Softwareprojekte typischen Kommunikationsmittel genannt werden. Abschließend wird der Onlinedienst SourceForge.net, welcher die zuvor beschriebenen Werkzeuge an zentraler Stelle vereint, vorgestellt.

### 3.3.1 Kommunikationsmittel

Die dominierenden Kommunikationsmittel für die weltweit verteilte Kooperation in freien Softwareprojekten sind<sup>99</sup>:

<sup>97</sup> Vgl. Biethahn (1996), S. 196 und Mertens/Bodendorf/König/Picot/Schumann (1996), S. 170.

<sup>98</sup> Vgl. Eilebrecht (2001) und Glasl (2001).

<sup>99</sup> Vgl. z.B. Lehmann (2001), Ronneburg (2001) oder Dalheimer (2001).

- eMail bzw. Mailinglisten<sup>100</sup>
- Newsgroups
- Internet Relay Chat (IRC)<sup>101</sup>

Viele OSS-Projekte verfügen über zwei oder mehrere Mailinglisten<sup>102</sup>. Üblich ist die Einrichtung einer öffentlich zugänglichen Liste, welche hauptsächlich für Anwender gedacht ist, sowie einer geschlossenen Liste für die Mitglieder des Core Teams und ggf. weitere Softwareentwickler, die auf Anfrage in diese Liste eingetragen werden können<sup>103</sup>.

Ursache für die Einrichtung mehrerer Mailinglisten kann z.B. eine zu hohe *Signal-To-Noise Ratio* auf der öffentlichen Liste sein. In einem solchen Fall überwiegt unwichtige über nützliche und dem Fortschritt des Projekts dienende Diskussion, sodass die Kernentwickler eine interne Liste anlegen, auf der sie über grundlegende Fragen der Entwicklung diskutieren können<sup>104</sup>.

### 3.3.2 Concurrent Versions System

Mit zunehmender Verbreitung und Komplexität standen die Betreuer freier Softwareprojekte zunehmend vor der Problematik einer effizienten Integration zahlreicher Patches, die von den Anwendern erarbeitet und an die Maintainer gesendet wurden. Mit der Zeit benötigten mehr und mehr Projekte ein System, "das automatisch Beiträge annimmt und jeden auf dem aktuellen Stand der Änderungen des Quelltextes hält<sup>105</sup>." Darüber hinaus sollte es möglich sein, bereits in den Quellcode integrierte Änderungen bei Bedarf – z.B. weil sie sich als fehlerhaft erwiesen haben – automatisch wieder zu entfernen. Das gesuchte System musste somit eine Art *Historie* des Projekts führen, um ältere Versionen wiederherstellen zu können<sup>106</sup>.

Walter Tichy schrieb mit dem *Revision Control System* (RCS) ein Programm, dass den o.g. Anforderungen weitgehend gerecht werden konnte. Aufgrund einiger

<sup>100</sup> Mailinglisten dienen der Kommunikation via eMail. Auf einem sog. Listserver werden die eMail-Adressen aller eingetragenen Nutzer abgelegt. Der Listserver selbst ist über eine feste eMail-Adresse erreichbar. Sobald einer der eingetragenen Nutzer eine eMail an den Listserver sendet, leitet dieser die Nachricht an alle anderen eingetragenen Nutzer weiter. Vgl. Institute for Telecommunication Sciences (Hrsg.) (2001).

<sup>101</sup> Auf IRC greifen nur wenige freie Softwareprojekte, wie z.B. The GIMP, zurück.

<sup>102</sup> Zu den Mailinglisten des KDE-Projekts gehören bspw. eine Liste für Anwender, auf der sich diese untereinander helfen können, eine Liste für Programmierer für Fragen hinsichtlich Entwurf und Entwicklung, eine Liste für Autoren und Übersetzer der Dokumentation, eine Liste für Lizenzfragen und eine geschlossene Liste für das Core Team, auf der bspw. über Fragen des Marketing und über die Öffentlichkeitsarbeit diskutiert wird. Vgl. Dalheimer (2001).

<sup>103</sup> Vgl. Lehmann (2001).

<sup>104</sup> Vgl. Lehmann (2001). Gleichermaßen kann die Bildung einer *offiziellen* Projektleitung auf derartige Ursachen zurückzuführen sein, vgl. Bezroukov (2001).

<sup>105</sup> Fogel (2000), S. 27 f.

<sup>106</sup> Vgl. Fogel (2000), S. 29 und S. 39.

Unzulänglichkeiten, wie z.B. fehlender Netzwerkfähigkeit, begannen jedoch andere Entwickler, RCS zu erweitern und später vollkommen neu zu schreiben. Ergebnis dieser Aktivitäten ist das Versionsverwaltungssystem *Concurrent Versions System* (CVS)<sup>107</sup>.

Mit dem CVS ist es nun möglich, die gleichzeitige Arbeit mehrerer Entwickler an denselben Dateien zu koordinieren und zusammenzufügen sowie ggf. über mögliche *Konflikte*<sup>108</sup> zu informieren. Aufgrund der Netzwerkfähigkeit können darüber hinaus Programmierer weltweit auf den Quellcode zugreifen.

Konkret können sich interessierte Entwickler mittels CVS die aktuellste Version (*Entwicklerversion*) aus dem sog. *CVS-Repository*<sup>109</sup> auf ihre lokalen Rechner kopieren, Veränderungen an Teilen des Quellcodes vornehmen und sich von CVS einen Patch erzeugen lassen, welcher dann – sofern keine Konflikte vorliegen – automatisch in das CVS-Repository integriert wird.

Die technische Umsetzung erfolgt derart, dass sich ein Entwickler mittels eines *Check Out* die Entwicklerversion als sog. *Arbeitskopie* auf seinen Rechner überträgt. Nach erfolgten Änderungen kann der Entwickler versuchen, diese durch ein *Commit* in das CVS-Repository einzuspielen. Erkennt CVS durch einen Abgleich der aktuellen Version mit der Arbeitskopie des Entwicklers, dass in der Zwischenzeit andere Programmierer Veränderungen am CVS-Repository vorgenommen haben, veranlasst es den Entwickler, vor dem Commit eine Aktualisierung (*Update*) seiner veralteten Arbeitskopie mit der Entwicklerversion durchzuführen. Sollte ein Konflikt zwischen der Entwicklerversion und der Arbeitskopie vorliegen, so muss der Entwickler die Ursache hierfür zunächst in seiner Arbeitskopie beheben, um einen Commit durchführen zu können.

Um unterschiedlich stark veraltete Arbeitskopien mehrerer Entwickler mit dem aktuellen CVS-Repository aktualisieren zu können, zeichnet CVS alle Commits seit Projektbeginn auf und aktualisiert die jeweiligen Arbeitskopien nur mit denjenigen Commits, die seit dem Check Out durchgeführt wurden<sup>110</sup>.

CVS fand vor allem deshalb einen hohen Verbreitungsgrad, weil sich durch intelligentes Zusammenführen von Veränderungen “die Entwickler nur selten um die logistischen Probleme kümmern [mussten], die entstehen, wenn mehrere Leute an den gleichen Quelltexten arbeiten<sup>111</sup>.” CVS reduziert somit den Aufwand sowohl für die

<sup>107</sup> Vgl. Fogel (2000), S. 29 f.

<sup>108</sup> Ein sog. *Konflikt* liegt vor, wenn zwei oder mehrere Entwickler gleichzeitig unterschiedliche Veränderungen an ein und derselben Stelle des Quellcodes vornehmen, vgl. Fogel (2000), S. 41.

<sup>109</sup> *CVS-Repository* ist in etwa mit *CVS-Archiv* zu übersetzen. Es handelt sich hierbei um den “Ort, an dem die Hauptquelltexte mit den Veränderungshistorien aufbewahrt werden.” Fogel (2000), S. 30.

<sup>110</sup> Eine detaillierte Beschreibung der Funktionalität des CVS findet sich in Fogel (2000).

<sup>111</sup> Fogel (2000), S. 30.

Betreuung eines Projekts als auch für die Mitarbeit an diesem, indem es Funktionen zum automatischen Erzeugen von Beiträgen in Form von Patches anbietet und öffentlichen Zugriff auf den Quellcode ermöglicht. Es ist demnach gleichermaßen ein Werkzeug für die Zusammenarbeit von Entwicklern und für die Softwareverteilung<sup>112</sup>.

### 3.3.3 SourceForge.net

SourceForge.net<sup>113</sup> ist ein freier Onlinedienst (*Hosting Service*) für Open Source Software-Entwickler, welcher von einer Gruppe von Anhängern freier Software betrieben und von VA Linux Systems unterstützt wird. Er bietet Softwareentwicklern verschiedenartige Hilfsmittel für die Verwaltung und die Betreuung ihres Projekts sowie Interessierten unterschiedliche Möglichkeiten der Beteiligung. Zu den zur Verfügung stehenden Mitteln zählen bspw.:

- CVS-Server für eigene CVS-Repositories
- diverse Administrations-Tools, z.B. für die Verwaltung der Rechte der Projektteilnehmer
- Projekt-Homepage in der Form <http://projektname.sourceforge.net>
- Mailingliste(n)
- Möglichkeiten des Dateitransfers via HTTP oder FTP
- MySQL-Datenbank
- verschiedene Tools für die Einreichung von Bug Reports, Patches und Feature Requests
- *Compile Farm*, welche den Entwicklern Zugriff auf verschiedenartige Rechnerarchitekturen und Betriebssysteme gewährt

SourceForge.net vereint nicht nur verschiedene Hilfsmittel für den Informationsaustausch und die Organisation eines freien Softwareprojekts, sondern bietet darüber hinaus mit CVS, einer Datenbank und der Möglichkeit, Software auf unterschiedliche Plattformen zu portieren, wichtige Werkzeuge für den eigentlichen Software-Entwicklungsprozess.

Dienste wie SourceForge.net<sup>114</sup>, die eine umfangreiche DV-technische Infrastruktur für freie Softwareprojekte an zentraler Stelle vereinen, können eine wesentliche

---

<sup>112</sup> Vgl. Fogel (2000), S. 143.

<sup>113</sup> Vgl. VA Linux Systems (Hrsg.) (2001a).

<sup>114</sup> Neben SourceForge.net gibt es noch weitere, z.T. kommerzielle Onlinedienste mit einem vergleichbaren Leistungsumfang. Vgl. hierzu Abschnitt 4.2.

Erleichterung für sämtliche Beteiligten eines solchen Projekts darstellen und einen beschleunigten Projektverlauf begünstigen<sup>115</sup>.

### 3.4 Modell des Verlaufs eines Open Source Software-Projekts

Der Ablauf kommerzieller Softwareprojekte besteht i.d.R. aus einer Folge systematisch geplanter Aktivitäten. Zu den bekanntesten Software Engineering-Modellen, die einen solchen Verlauf abbilden, zählt das sog. *Wasserfall-Modell*, welches die Vorgehensweise der Softwareentwicklung als das kaskadenartige Durchlaufen einer Reihe klar definierter Phasen beschreibt<sup>116</sup>. Wie jedoch in Abschnitt 3.2.2.1 bereits dargestellt wurde, lässt sich auf freie Softwareprojekte keines der üblichen Phasenkonzepte gänzlich übertragen<sup>117</sup>.

Der eigentliche Prozess der Softwareentwicklung innerhalb einer Open Source-Community dagegen ähnelt der Methode des evolutionären Prototyping (*Rapid Prototyping*). Auch in freien Softwareprojekten werden laufend – aufgrund von Patches, Bug Reports und Feature Requests – Modifikationen an der getesteten Software vorgenommen, bis diese die geforderte bzw. gewünschte Funktionalität erreicht hat. Beiden Vorgehensweisen gemeinsam ist somit die Tatsache, dass in einem kontinuierlichen Prozess eine erste, möglichst frühzeitig realisierte und vorgelegte Version schrittweise erweitert wird<sup>118</sup>.

Diejenigen Ereignisse und Tätigkeiten, welche sich als typisch für freie Softwareprojekte und insbesondere für deren Software-Entwicklungsprozess herausgestellt haben, sollen im folgenden modellartig dargestellt werden. Dieses Modell soll und kann jedoch kein Referenzmodell für die OSS-Entwicklung darstellen, sondern lediglich den für derartige Projekte üblichen Verlauf illustrieren<sup>119</sup>.

<sup>115</sup> Obwohl SourceForge.net erst im November 1999 gegründet wurde, beläuft sich die Anzahl der hierüber koordinierten OSS-Projekte zum 2001-10-17 auf 28.039. 273.566 registrierte User wurden an diesem Tag gezählt. SourceForge.net gilt als der führende Application Service Provider (ASP) für OSS-Entwickler weltweit.

<sup>116</sup> Vgl. Ghezzi/Jazayeri/Mandrioli (1991), S. 360 - 374.

<sup>117</sup> Davon abgesehen würde die Anwendung eines Phasenkonzepts aufgrund des Umfangs und der Komplexität zahlreicher freier Softwareprojekte einen unverhältnismäßigen Mehraufwand bedeuten.

<sup>118</sup> Vgl. Biethahn/Mucksch/Ruf (1996), S. 213 - 216, Yeh (1993), S. 8 und Fogel (2000), S. 218 ff.

<sup>119</sup> Grundlage dieser modellartigen Darstellung sind die bisherigen Ausführungen sowie vor allem Raymond (2001d) und Fogel (2000), S. 125 - 128, weshalb auf weitere Erläuterungen hierzu an dieser Stelle verzichtet werden soll.



### 3.4.1 Ereignisgesteuerte Prozessketten

Zur Darstellung des Projektverlaufs wird auf die Methode der *Ereignisgesteuerten Prozessketten* (EPK) zurückgegriffen. Nachfolgend sollen die wesentlichen Merkmale dieser Methode kurz erläutert werden<sup>120</sup>.

EPK haben die Darstellung der zeitlich-logischen Abfolge von Funktionen zum Gegenstand. Über das Konzept der Ereignissteuerung können dynamische Prozesse abgebildet werden. Funktionen und Ereignisse, die durch logische Verknüpfungsoperatoren zueinander in Beziehung gebracht werden, stellen demnach die wesentlichen Elemente für eine Prozessmodellierung dar.

#### Funktion

Eine Funktion im Sinne einer Aufgabe stellt eine durch physische oder geistige Aktivitäten zu verwirklichende Sollleistung dar. Funktionen werden durch Ereignisse ausgelöst.

#### Ereignis

Unter einem Ereignis wird der Eintritt eines definierten Zustands, welcher eine Folge von Aktivitäten bewirken kann, verstanden. Ereignisse können demnach Funktionen auslösen.

#### Verknüpfung

Mögliche Verknüpfungsarten sind:

- konjunktive Verknüpfung (*und*-Verknüpfung): Sind zwei Aussagen konjunktiv miteinander verknüpft, dann ist die Gesamtaussage wahr, wenn die beiden miteinander verknüpften Aussagen wahr sind.
- disjunktive Verknüpfung (*entweder oder*-Verknüpfung): Eine disjunktive Verknüpfung von zwei Aussagen besagt, dass die Gesamtaussage wahr ist, wenn genau eine Aussage wahr ist.
- adjunktive Verknüpfung (*und/oder*-Verknüpfung): Bei einer adjunktiven Verknüpfung zweier Aussagen ist die Gesamtaussage wahr, wenn mindestens eine Aussage wahr ist.

Abbildung 3.4.1/1 zeigt die zur Darstellung der Elemente benutzten Symbole.

---

<sup>120</sup> Vgl. zu den weiteren Ausführungen über Ereignisgesteuerte Prozessketten u.a. Scheer (1992) und Keller/Nüttgens/Scheer (2001).

Abb. 3.4.1/1: Symbole zur Darstellung der Elemente einer EPK

### 3.4.2 Modell eines Projektverlaufs

In Abbildung 3.4.2/1 erfolgt zunächst die Darstellung derjenigen Ereignisse und Tätigkeiten, welche gewöhnlich zum Beginn, zur Übernahme von oder zur Teilnahme an freien Softwareprojekten führen<sup>121</sup>.

Abbildung 3.4.2/2 zeigt den typischen Verlauf des eigentlichen Software-Entwicklungsprozesses aus der Sicht eines sich nicht nur auf die Verwaltung und Koordination des Projekts beschränkenden Maintainers.

---

<sup>121</sup> Alternative Ereignisse und Tätigkeiten, die nicht in diesen Entscheidungen resultieren, bleiben unberücksichtigt.



Abb. 3.4.2/2: Typischer Verlauf des Software-Entwicklungsprozesses

# Kapitel 4

## Ökonomische Implikationen

Nach anfänglichen Vorbehalten zeigten in den vergangenen Jahren mehr und mehr Unternehmen der Computerindustrie ein gesteigertes Interesse an Open Source Software (OSS). Unternehmen wie Hewlett-Packard und IBM – welches zudem im Sommer 2000 ankündigte, jeweils mehr als \$200 Mio. in GNU/Linux-Entwicklungszentren in Europa und Asien zu investieren – haben sich bereits auf das Experiment einer OSS-Entwicklung eingelassen. Darüber hinaus konnte vor allem in der jüngsten Vergangenheit beobachtet werden, dass verschiedene Unternehmen freie Software einsetzen oder ihr Geschäftsmodell<sup>1</sup> um freie Software ergänzen bzw. auf ihr aufbauen<sup>2</sup>. Selbst die Europäische Union hat die Förderung von OSS empfohlen sowie die Bevorzugung freier Software gegenüber proprietären Produkten bei Ausschreibungen in Aussicht gestellt. Die Konzepte der Community wurden schließlich nicht nur von Managern, sondern gleichermaßen von Analysten und Investoren untersucht<sup>3</sup>.

Die größere Beachtung einer breiteren Öffentlichkeit sowie die zunehmende Akzeptanz und Verbreitung freier Software deuten darauf hin, dass die Aktivitäten und Methoden der Open Source-Community für die Computer- und insbesondere für die Softwareindustrie nicht folgenlos bleiben werden.

Bevor in diesem Zusammenhang ein Überblick über alternative Konzepte, welche über die herkömmliche Vermarktung von Software hinausgehen, gegeben wird, sollen zunächst mögliche Auswirkungen freier Software auf bestehende Geschäftsmodelle beleuchtet werden. Abschließend wird untersucht, ob und inwieweit für Unternehmen die Möglichkeit besteht, sich an freien Softwareprojekten zu beteiligen und inwiefern

---

<sup>1</sup> Der Begriff *Geschäftsmodell* wird nicht einheitlich definiert. Im folgenden sollen unter diesem Begriff unternehmensspezifische Prozesse und Leistungen im Rahmen einer auf Gewinnerzielung gerichteten Tätigkeit verstanden werden.

<sup>2</sup> Jonathan L. Prial, IBM Director Integrated Solutions and Linux Marketing, begründet die Unterstützung von GNU/Linux durch IBM folgendermaßen: “Wir glauben, ... dass dieser Markt abheben und sich zu einer ernsthaften Alternative für Kunden entwickeln wird. Und wir sind entschlossen, früh einzusteigen und mit Linux zu wachsen, so wie Linux selbst wächst.” Vgl. Weber (2001).

<sup>3</sup> Vgl. Behlendorf (1999), S. 149, Hetze (2001) und BMWi (Hrsg.) (2001), S. 6 und S. 30.

organisatorische Aspekte des Open Source-Modells auf Unternehmen übertragbar sind.

## 4.1 Auswirkungen auf bestehende Geschäftsmodelle

Die Betrachtung möglicher Folgen für bestehende Geschäftsmodelle aufgrund eines zunehmenden Erfolgs freier Software erfordert nahezu zwangsweise den Vergleich mit proprietärer Software bzw. mit traditionellen, kommerziell orientierten Softwareunternehmen. In diesem Zusammenhang sollen nach der Darstellung anwendungsbezogener Vorteile von OSS sowohl erfolgversprechende als auch nachteilige Maßnahmen bzw. Folgen für Unternehmen der Hard- und Softwareindustrie sowie deren Ursachen untersucht werden.

### 4.1.1 Vorteile und Chancen durch Open Source Software

#### 4.1.1.1 Vorüberlegungen

Viele Unternehmen betrachten die von ihnen entwickelte und vermarktete Software als ihr geheimes, geistiges Eigentum (*intellectual property*), das es unbedingt vor einer unbefugten Einsichtnahme zu schützen gilt<sup>4</sup>. Die durch proprietäre Software angestrebte Wahrung des geistigen Eigentums verhindert jedoch weitgehend jede Unterstützung durch externe Anwender und Entwickler<sup>5</sup>.

Kann sich die Unternehmensleitung dagegen von dieser Überzeugung lösen und die Unterstützung der Open Source-Community gewinnen, so ergeben sich verschiedene Strategien, die aufgrund der nunmehr möglichen verteilten, offenen Softwareentwicklung verfolgt werden können. Insbesondere bietet sich für Softwareunternehmen die Möglichkeit, durch die Veröffentlichung ihrer Produkte unter einer OSS-Lizenz die Fähigkeiten der Community zur Verbesserung und Erweiterung ihrer Software zu nutzen, wodurch Kosten für Entwickler und externe Tester entfallen oder wenigstens reduziert werden können<sup>6</sup>.

In diesem Zusammenhang äußern jedoch Vertreter der Softwareindustrie vereinzelt Bedenken, dass, sobald sie den Quellcode ihrer Produkte unter einer Lizenz wie der GPL veröffentlichen würden, Wettbewerber jederzeit darauf aufbauend eigene,

---

<sup>4</sup> Craig Mundie, Microsofts Senior Vice President of Advanced Strategies, vertritt hierzu bspw. die Ansicht, dass ohne den Schutz geistigen Eigentums keine Einkünfte zu erzielen, somit keine Forschung zu finanzieren und letztendlich keine Innovationen zu erwarten seien. Vgl. Mundie (2001).

<sup>5</sup> Vgl. Lerner/Tirole (2000), S. 25 und BMWi (Hrsg.) (2001), S. 7.

<sup>6</sup> Vgl. Lerner/Tirole (2001), S. 822. Es handelt sich hierbei um eine Form des Outsourcing. Von entscheidender Bedeutung ist jedoch, das Vertrauen der Open Source-Community zu gewinnen. Sowohl die Formulierung der Lizenzbestimmungen als auch die Art und Weise der Projektleitung sollten wohl überlegt sein.

proprietäre Lösungen entwickeln und vermarkten könnten. Dem entgegen steht jedoch die Tatsache, dass aufgrund der Lizenzbestimmungen die Verpflichtung bestehen kann, abgeleitete Werke wiederum unter derselben Lizenz zu veröffentlichen. Demnach stehen diese abgeleiteten Werke gleichermaßen den Entwicklern der ursprünglichen Software zur Verfügung. Ein solcher Wettbewerber befindet sich schließlich in unmittelbarer Konkurrenz zu den ursprünglichen Autoren und Produzenten, denen i.d.R. eine größere Kompetenz zugetraut wird<sup>7</sup>.

Diese Überlegungen lassen jedoch bereits erkennen, dass sich Unternehmen, die ihr Geschäftsmodell um freie Software ergänzen bzw. auf ihr ausrichten, in einem Netz gegenseitiger Abhängigkeiten aus traditionellen Geschäftskonzepten, freien Softwarelizenzen und finanziellen Erfordernissen befinden.

#### 4.1.1.2 Anwendungsbezogene Vorteile von Open Source Software

Vieles deutet darauf hin, dass freie Software zunehmend im gewerblichen Bereich eingesetzt wird und an Bedeutung gewinnen kann. Untersuchungen kommen bspw. zu dem Ergebnis, dass allein GNU/Linux auf dem Servermarkt gegenüber Microsoft mithalten kann. Allgemein wird für GNU/Linux in den kommenden Jahren eine durchschnittliche Wachstumsrate von 200% angenommen<sup>8</sup>.

Neben ihrer kostenlosen Verfügbarkeit und dem zunehmenden Angebot an OSS scheint ihr Einsatz mit weiteren Vorteilen verbunden zu sein. Die meistgenannten Ursachen für die zunehmende Nutzung freier Software über den privaten und universitären Bereich hinaus lassen sich unter den folgenden Punkten zusammenfassen.

#### Überprüfbare Sicherheit

Vertreter kommerzieller Softwareunternehmen weisen oftmals darauf hin, dass der Einsatz proprietärer Software aufgrund der Geheimhaltung des Quellcodes den besten Schutz gegen mögliche Angriffe von Crackern bieten könne<sup>9</sup>. Inzwischen setzt sich jedoch – nicht zuletzt aufgrund des Eindrucks, dass *Software-Monokulturen* für Angriffe besonders anfällig sind – zunehmend die Erkenntnis durch, dass durch die Verfügbarkeit des Quellcodes die Sicherheit des Systems am zuverlässigsten überprüft werden kann. Freie Software gewinnt durch die Möglichkeit, jederzeit jedes Detail nachprüfen zu können, zunehmend das Vertrauen nicht zuletzt gewerblicher Anwender. Selbst Regierungen denken inzwischen darüber nach, ihre Verwaltungen mit OSS auszustatten, da sie nicht ausschließen können, dass bei proprietärerer Software Unbefugten der Zugang zu sensiblen Daten ermöglicht wird<sup>10</sup>.

---

<sup>7</sup> Vgl. Behlendorf (1999), S. 168.

<sup>8</sup> Vgl. Bayer (2001), Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik (KBSt) (Hrsg.) (2001) und Lerner/Tirole (2000), S. 1.

<sup>9</sup> Vgl. bspw. Mundie (2001).

<sup>10</sup> Vgl. BMWi (Hrsg.), S. 3 und S. 20 f. sowie Köppen/Nüttgens (2000), S. 232.

### Geringe Kosten

Insbesondere diejenigen Anwender, welche die Lizenzgebühren proprietärer Software nicht oder nur schwer aufbringen können oder wollen, verfügen mit freier Software i.d.R. über eine gleichwertige, kostenlose und beliebig oft reproduzierbare Alternative zu proprietären Programmen. Das Fehlen detaillierter Dokumentationen oder einfach zu bedienender Benutzerschnittstellen wird ggf. für eine Kostenersparnis in Kauf genommen<sup>11</sup>. Vor allem in als weniger entwickelt geltenden Ländern können sowohl private als auch gewerbliche Anwender von freier, kostenloser Software profitieren<sup>12</sup>. Zudem können ihre weitgehende Plattformunabhängigkeit sowie der häufig erwähnte ressourcenschonende Umgang freier Software mit der Hardware ggf. eine längere Nutzung derselben ermöglichen bzw. ihre Anpassung erübrigen und somit zu einem weiteren Kostenvorteil führen<sup>13</sup>.

### Herstellerunabhängigkeit

Freie Software hat keinen Eigentümer, welcher allein über die Verifizierung, die Verfügbarkeit und die Weiterentwicklung der Software entscheiden kann. Sie ist unabhängig von Produktzyklen, Vermarktungsstrategien und Insolvenzen einzelner Unternehmen. Sofern ein OSS-Projekt offensichtlich nicht mehr fortgeführt wird, kann ein Unternehmen bei Bedarf jederzeit selbst notwendige Anpassungen vornehmen. Das aufgrund des Einsatzes proprietärer Software bestehende Risiko der Abhängigkeit von den Entscheidungen des Softwareproduzenten sowie die Möglichkeit einer damit einhergehenden Reduzierung der Investitionssicherheit ist bei freier Software demnach vergleichsweise gering<sup>14</sup>.

### Verbreitung offener Standards

Die Spezifikationen zahlreicher Protokolle und Formate sind uneingeschränkt zugänglich als offene Standards in den sog. *Request For Comments* (RFC) der Internet Engineering Task Force (IETF) – einer offenen und zentralen Standardisierungs-Organisation des Internet – beschrieben<sup>15</sup>. Die Tatsache, dass sich in einem lukrativen Bereich wie dem für lokale Netzwerke die freien Internetprotokolle gegen proprietäre Ansätze – z.B. gegen das DECnet und die Systems Network Architecture (SNA) von IBM – durchsetzen konnten, unterstreicht die Bedeutung offener Standards. Weitere Standardisierungen lizenzgebührenfreier Softwareschnittstellen und -formate, welche

<sup>11</sup> Vgl. Lerner/Tirole (2000), S. 8.

<sup>12</sup> Vgl. Lerner/Tirole (2000), S. 2 und Kahney (2001).

<sup>13</sup> Vgl. BMWi (Hrsg.) (2001), S. 13 und S. 25 sowie (KBSt) (Hrsg.) (2001).

<sup>14</sup> Vgl. Köppen/Nüttgens (2000), S. 232, BMWi (Hrsg.) (2001), S. 45 und KBSt (Hrsg.) (2001). Die KBSt kommt bspw. zu dem Ergebnis, dass "...durch die große Resonanz im Umfeld der IT-Industrie ... Linux heutzutage einen guten Investitionsschutz [bietet]."

<sup>15</sup> Die Urheberrechte an den Request For Comments und ihren Drafts selbst liegen seit 1994 bei der Internet Society, einer internationalen Koordinierungsstelle des Internet.



zudem die Entwicklung zueinander kompatibler Systeme begünstigen und die Portierung auf andere Plattformen ermöglichen, können demnach wesentlich dazu beitragen, dass die Bedeutung proprietärer Softwarelösungen tendenziell abnehmen wird<sup>16</sup>.

#### 4.1.1.3 Möglichkeiten für Unternehmen der Computerindustrie

Die genannten Vorteile kommen in erster Linie für die Anwender freier Software zum Tragen. Sofern jedoch Unternehmen der Computerindustrie dazu bereit sind, ihre Produkte bzw. Informationen darüber zu veröffentlichen, können sich für sie verschiedene Möglichkeiten durch die Einbeziehung und Unterstützung der Open Source-Community eröffnen.

Die Ausführungen des folgenden Abschnitts basieren teilweise auf der Annahme, dass sich aufgrund der zu beobachtenden, immer kürzer werdenden Produktzyklen selbst durch eine umfassende unternehmensinterne Kontrolle des Software-Entwicklungsprozesses nicht gänzlich sicherstellen lässt, dass kommerzielle Software fehlerfrei oder termingerecht veröffentlicht wird<sup>17</sup>. Zudem kann es für Unternehmen überlegenswert sein, die Weiterentwicklung ihrer Software der Open Source-Community zu überlassen, da Änderungen und Erweiterungen bestehender Software ohnehin den größten Teil der herkömmlichen Softwareentwicklung ausmachen<sup>18</sup>.

#### Offenlegung von Systemspezifikationen

Eine in erster Linie für Unternehmen der Hardwareindustrie denkbare Strategie basiert auf der umgehenden Veröffentlichung der Spezifikationen und Details ihrer Komponenten.

Die Entwicklung hardwarenaher Software – angefangen von Gerätetreibern über Konfigurationssoftware bis zu kompletten Betriebssystemen – ist oftmals mit hohen Kosten verbunden, gleichzeitig lassen sich mit ihr nahezu keine unmittelbaren Erlöse erzielen<sup>19</sup>. Aus diesen Gründen kann es für Unternehmen überlegenswert sein, die (Weiter-)Entwicklung derartiger Software der Open Source-Community zu überlassen. Neben einer möglichen Reduzierung der Entwicklungskosten bei nahezu gleichbleibenden Erlösen können Kunden auch nach Einstellung einer offiziellen Unterstützung seitens des Produzenten (z.B. aufgrund technischen Fortschritts) Änderungen und Erweiterungen der Software selbst vornehmen, wodurch eine höhere Kundenzufriedenheit und -loyalität begünstigt werden kann<sup>20</sup>.

<sup>16</sup> Vgl. Köppen/Nüttgens (2000), S. 240 und Hetze (2001). Das BMWi empfiehlt bspw. die Verwendung offener Standards für die Datenarchivierung und den Datenaustausch in heterogenen Netzen, vgl. BMWi (Hrsg.) (2001), S. 44 f.

<sup>17</sup> Vgl. Hetze (2001).

<sup>18</sup> Der Aufwand für Wartung und Pflege ist gewöhnlich um einen Faktor von zwei bis vier höher als der Aufwand für die eigentliche Softwareentwicklung. Vgl. Balzert (2001), S. 1093 f.

<sup>19</sup> Vgl. Raymond (2001g), S. 135 f. und S. 163 - 166.

<sup>20</sup> Vgl. Hecker (2001) und Lerner/Tirole (2000), S. 33 f.

Apple Computer bspw. veröffentlichte Anfang 1999 *Darwin*, den Kernel des Mac OS X-Betriebssystems, unter der Apple Public Source License<sup>21</sup>.

### Nachträgliche Freigabe als Open Source Software

Eine mit der o.g. vergleichbare Strategie besteht in der nachträglichen Veröffentlichung ehemals proprietärer Software unter einer OSS-Lizenz. Sobald zu erwarten ist, dass die durch eine Weiterentwicklung der Software innerhalb der Open Source-Community reduzierten Kosten die geringeren Erlöse durch entgangene Lizenzgebühren übersteigen, kann sich für ein Unternehmen die Gelegenheit bieten, seine Software unter einer OSS-Lizenz freizugeben. Darüber hinaus sind ebenfalls positive Auswirkungen auf die Kundenzufriedenheit und -loyalität denkbar.

Sollte sich ein Unternehmen für die nachträgliche Freigabe seiner Software entscheiden besteht zudem ggf. die Möglichkeit, entgangene Erlöse durch ergänzende Produkte oder technische Unterstützung auszugleichen<sup>22</sup>.

In diesem Zusammenhang ist jedoch anzunehmen, dass der Erfolg einer solchen Maßnahme wesentlich vom Zeitpunkt der Freigabe abhängen kann. Eine zu späte Veröffentlichung wird i.d.R. nur noch wenige prestigeträchtige Tätigkeiten bieten, sodass sich interessierte Softwareentwickler mitunter einem anderen Projekt zuwenden bzw. ihre Beteiligung daran fortsetzen<sup>23</sup>. Darüber hinaus kann eine bereits weit fortgeschrittene Software ein erhöhtes Maß an Komplexität erreicht haben, dessen Verständnis bzw. Überwindung für interessierte Teilnehmer einen zu hohen Aufwand bedeuten würde und sie von einer Beteiligung absehen lassen kann. Gleichermäßen kann die zu frühe Veröffentlichung einer noch nicht funktionsfähigen Version potenzielle Entwickler von einer Beteiligung abhalten<sup>24</sup>.

Beispiele für Unternehmen, die ihre ehemals proprietäre Software unter einer OSS-Lizenz freigegeben haben, sind Inprise/Borland, dessen Datenbank *InterBase* im Juli 2000 der InterBase Public License – einer Variante der Mozilla Public License – unterstellt wurde sowie die SAP AG, welche die verschiedenen Komponenten ihrer Datenbank *SAP DB* im Oktober 2000 unter der GPL bzw. der LGPL freigab<sup>25</sup>.

### Mehrfachlizenzierung

Unternehmen der Softwareindustrie können ihre Produkte aufgrund der ihnen zu-

<sup>21</sup> Vgl. Apple Computer (Hrsg.) (2001).

<sup>22</sup> Vgl. Hecker (2001) und Lerner/Tirole (2000), S. 29 f.

<sup>23</sup> Üblicherweise erweisen sich diejenigen OSS-Entwickler, welche zu den ersten Teilnehmern eines Projekts zählen, als ausgesprochen loyal gegenüber dem Projekt und seinem Betreuer, vgl. Bezroukov (2001).

<sup>24</sup> Vgl. Lerner/Tirole (2000), S. 28 und Bezroukov (2001).

<sup>25</sup> Vgl. Borland Software (Hrsg.) (2001a) und (2001b) sowie SAP (Hrsg.) (2001).

stehenden Urheberrechte gleichzeitig unter verschiedenen Lizenzen veröffentlichen<sup>26</sup>. Eine kommerzielle Nutzung der entsprechenden Software kann bspw. einer proprietären Lizenz unterliegen, wohingegen eine private oder nicht-kommerzielle Nutzung einer OSS-Lizenz unterstellt wird. Aufgrund einer solchen Mehrfachlizenzierung können Unternehmen einerseits Erlöse aus dem Verkauf proprietärer Lizenzen erzielen sowie andererseits auf eine Unterstützung durch die Open Source-Community hoffen<sup>27</sup>.

Da ein Softwareunternehmen, welches eine derartige Lizenzpolitik in Erwägung zieht, sicherstellen sollte, dass Vorschläge und Erweiterungen aus der Community gleichermaßen in die proprietäre Version integriert werden können, ist für dieses Modell die Formulierung der Lizenzbestimmungen von zentraler Bedeutung<sup>28</sup>.

Netscape Communications bspw. veröffentlichte den Netscape Communicator sowohl unter der Mozilla Public License (MPL) als auch unter der Netscape Public License (NPL), welche dem Unternehmen zusätzliche Rechte gewährt<sup>29</sup>. Ein weiteres Beispiel für ein Unternehmen, das seine Software unter zwei verschiedenen Lizenzen veröffentlicht, ist Aladdin Enterprises, dessen *Ghostscript* sowohl der GPL als auch der Aladdin Free Public License (AFPL) unterliegt<sup>30</sup>.

#### 4.1.2 Nachteile und Risiken durch Open Source Software

In der jüngsten Vergangenheit ließ sich zunehmend beobachten, dass Unternehmen, die ihr Geschäftsmodell auf freie Software im Allgemeinen oder GNU/Linux im Besonderen ausrichteten, verstärkt Maßnahmen ergreifen mussten, um ihren Geschäftsbetrieb aufrechterhalten zu können. Offensichtlich haben sich für manche Unternehmen im Open Source-Umfeld die zunächst hohen Erwartungen nicht im erhofften Maße erfüllt<sup>31</sup>.

Ein wesentlicher Grund hierfür scheint darin zu bestehen, dass derartige Unter-

---

<sup>26</sup> Eine vergleichbare Möglichkeit besteht darin, die Software unter einer einzigen Lizenz zu veröffentlichen, die jedoch bspw. für unterschiedliche Anwendungsbereiche abweichende Bestimmungen enthält.

<sup>27</sup> Vgl. Hecker (2001). In gewisser Weise ähnelt Microsofts *Shared Source Philosophy* diesem Modell. *Shared Source* besagt, dass der Quellcode ausgesuchten Geschäftspartnern zugänglich gemacht werden kann, jedoch sind weder Änderungen daran noch dessen Weiterverteilung erlaubt. Microsoft erklärt hierzu, dass aufgrund dieses Modells Interessierte jederzeit Hinweise auf Fehler und Verbesserungen der Software geben könnten bei gleichzeitigem Schutz des geistigen Eigentums des Unternehmens durch dessen alleiniges Recht, Änderungen und Weiterentwicklungen umzusetzen. Vgl. Mundie (2001).

<sup>28</sup> Vgl. Behlendorf (1999), S. 168. In diesem Zusammenhang besteht z.B. die Möglichkeit, bei Bedarf von einem Teilnehmer, der einen Beitrag für die freie Version geleistet hat, eine Erlaubnis einzuholen, welche die Integration seines Beitrags in die proprietäre Version ermöglicht.

<sup>29</sup> Vgl. The Mozilla Organisation (Hrsg.) (2001b) und BMWi (Hrsg.) (2001), S. 14. Die NPL erlaubt Netscape Communications bspw., Beiträge anderer Entwickler jederzeit ohne vorherige Erlaubnis in eine proprietäre Version zu integrieren.

<sup>30</sup> Vgl. Aladdin Enterprises (Hrsg.) (2001).

<sup>31</sup> Vgl. Bayer (2001).

nehmen keine Erlöse aus dem Verkauf freier Software gegen Lizenzgebühren erzielen können<sup>32</sup>. Im Gegensatz zu traditionellen Unternehmen der Softwareindustrie sind solche aus dem OSS-Umfeld in hohem Maße darauf angewiesen, andere Erlösquellen zu erschließen und ihre potenziellen Kunden von dem Konzept und den möglichen Nutzen freier Software zu überzeugen.

Im Zusammenhang mit freier Software lassen sich neben den entgangenen Lizenzgebühren allgemein die folgenden, häufig unvorhersehbaren Ereignisse und Nachteile für verschiedene Unternehmen ausmachen.

#### 4.1.2.1 Unternehmen im Open Source-Umfeld

##### **Fragliche Unterstützung durch die Open Source-Community**

Ein Unternehmen sollte keineswegs erwarten, dass sich die Open Source-Community umgehend und dauerhaft seines ehemals proprietären Produkts annahmen wird. Es ist zu bezweifeln, dass allein durch die Veröffentlichung der Software unter einer OSS-Lizenz die Unterstützung der Community erreicht werden kann. Vielmehr sollte das Produkt die Bedürfnisse der freien Entwicklergemeinschaft treffen und sie somit für eine Beteiligung motivieren. Das Unternehmen muss hierfür eine Form der Zusammenarbeit schaffen, die für alle Beteiligten vorteilhaft ist und von der beide Seiten profitieren können<sup>33</sup>.

##### **Lizenz- und haftungsrechtliche Problemstellungen**

Obwohl sich die Rechtsprechung bislang nur in geringem Umfang mit OSS beschäftigte und der Eindruck entstehen könnte, freie Software befinde sich in einem rechtsfreien Raum, unterliegt jede kommerzielle Nutzung derartiger Software vertraglichen bzw. gesetzlichen Bestimmungen, deren Verletzung erhebliche Folgen nach sich ziehen kann<sup>34</sup>. Darüber hinaus hat ein OSS-Unternehmen in seinen Handlungen die rechtlichen Beziehungen zwischen sich, seinen Kunden und dem Autor der Software zu beachten.

Aufgrund der geltenden Bestimmungen sind einige (wenige) Klauseln von OSS-Lizenzen nach deutschem Recht unwirksam. Demnach können bspw. die Folgen grob fahrlässiger Handlungen oder arglistigen Verhaltens im Zusammenhang mit einer

<sup>32</sup> Hans Bayer, Geschäftsführer der Caldera Deutschland GmbH, erläutert hierzu bspw., dass es nicht funktionieren könne, Profite zu erzielen, indem man Software praktisch verschenkt. Vgl. Bayer (2001).

<sup>33</sup> Vgl. O'Reilly (2001). Rishab Aiyer Ghosh vergleicht diese Form der Kooperation mit einem Tauschgeschäft, in dem beide Parteien der jeweils anderen etwas Wertvolles anbieten müssen. Da man das in einem Tauschgeschäft Erworbene üblicherweise benutze und nicht weiter tausche, wird ihm ein Wert zugeschrieben. Ein auf Tausch basierendes System funktioniere jedoch nur, so lange sich die korrespondierenden Wertflüsse gegenseitig ausbalancieren. Vgl. Ghosh (2001).

<sup>34</sup> In Deutschland gelten in diesem Zusammenhang vor allem die Bestimmungen des Bürgerlichen Gesetzbuchs (BGB), des Gesetzes zur Regelung des Rechts der Allgemeinen Geschäftsbedingungen (AGBG) und des Urheberrechtsgesetzes (UrhG).

kommerziellen Nutzung freier Software zu Schadensersatzansprüchen führen<sup>35</sup>.

#### 4.1.2.2 Traditionelle Softwareunternehmen

Aufgrund des zunehmenden Erfolgs freier Software sollten kommerzielle Softwareunternehmen jederzeit damit rechnen, dass die Open Source-Community ein vergleichbares und qualitativ gleichwertiges Produkt hervorbringen kann. Bei einem gleichzeitigen Imageverlust proprietärer Software kann zudem die Tatsache, dass freie Software mit – von Technikern oftmals geschätzter – UNIX-Technologie in Verbindung gebracht wird, die Akzeptanz und Verbreitung derartiger Software zusätzlich fördern<sup>36</sup>. Darüber hinaus kann sich OSS aufgrund ihrer uneingeschränkten Verfügbarkeit schnell und unbegrenzt verbreiten<sup>37</sup>.

Die Open Source-Community kann sich jederzeit zu einem ernstzunehmenden Wettbewerber für traditionelle Softwareunternehmen, welcher zudem durch herkömmliche kapitalmäßige Transaktionen nicht beeinflussbar ist<sup>38</sup>, entwickeln.

## 4.2 Open Source Software als Gegenstand der Geschäftstätigkeit

Die mit freier Software verbundenen Nutzungsrechte erlauben keinen Verkauf von Software gegen Lizenzgebühren, wie es für proprietäre Software üblich ist. Zudem kann derartige Software beliebig oft kopiert und weiterverteilt werden, ihre Reproduktion ist nur mit unwesentlichen Kosten verbunden. Da der Preis, den ein Anbieter von OSS verlangen kann, jedoch lediglich die Kosten der Vervielfältigung und Bereitstellung ausgleichen darf, sind mit freier Software demnach keine nennenswerten, direkten Erlöse erzielbar.

Aufgrund der gestiegenen Nachfrage hat sich dennoch ein Markt für derartige Software gebildet. Eine Unternehmung kann hier verschiedene Strategien verfolgen, um indirekt Erlöse durch und mit OSS zu erzielen. Die bekanntesten derartigen, sich teilweise überschneidenden Konzepte sollen im folgenden anhand von Beispielen dargestellt werden.

---

<sup>35</sup> Mit einer zunehmenden wirtschaftlichen Bedeutung freier Software ist daher vermehrt mit rechtlichen Auseinandersetzungen zu rechnen. Aufgrund der Komplexität dieses Themas soll jedoch an dieser Stelle auf weitere Ausführungen hierzu verzichtet werden. Detaillierte Informationen hinsichtlich vertraglicher und rechtlicher Probleme bei der kommerziellen Nutzung freier Software finden sich in Siepmann (2001).

<sup>36</sup> Vgl. Hetze (2001). Die zunehmende Verbreitung freier Software in den unterschiedlichsten Unternehmen ist nicht zuletzt auf die Initiative von DV-Technikern zurückzuführen.

<sup>37</sup> Vgl. Hetze (2001).

<sup>38</sup> Vgl. Abschnitt 4.3.

“ d” wud Zub”

N N

N N N

(Accsss)

, &

, &

N

K u

die Möglichkeit, durch Ausrichtung ihrer Kompetenzen auf OSS-spezifische Lösungen als ein hierauf spezialisierter IT-Dienstleister aufzutreten. Strategische Kooperationen zwischen dem Beratungsunternehmen und dessen Kunden zur gemeinsamen Entwicklung problemspezifischer Lösungen auf der Basis freier Software können zudem eine langfristige Kundenbindung begünstigen. Schließlich ist mit einer verstärkten Ausrichtung auf freie Software eine verminderte Abhängigkeit von Produzenten proprietärer Software zu erwarten<sup>44</sup>.

Cygnus Solutions bspw. wurde 1989 als erstes und damit dieses Geschäftsmodell etablierendes Unternehmen gegründet. Trotz Zweifeln hatte das Unternehmen mit dem Konzept, technische Unterstützung für z.T. selbst entwickelte<sup>45</sup>, freie und nicht zuletzt kostenlose Software anzubieten, Erfolg. Fünf Jahre nach Gründung verzeichnete Cygnus Solutions, zu dessen Kunden u.a. Intel, Hewlett-Packard, Corel und Oracle gehörten, ein Auftragsvolumen von \$5,7 Mio., das Forbes Magazine bezeichnete im August 1998 das Unternehmen als größtes Open Source-Unternehmen weltweit<sup>46</sup>. Anfang 2000 wurde Cygnus Solutions für \$674 Mio. von Red Hat übernommen.

Neben Red Hat bieten u.a. IBM, VA Linux Systems und die Innominate AG kommerziellen Support für freie Software an.

## Ergänzende Dienst- und Zusatzleistungen

Zu den bekanntesten Unternehmen bzw. Organisationen, deren Geschäftsgrundlage vor allem in ergänzenden, softwarebezogenen Leistungen zu freier Software besteht, zählen die sog. Distributoren, welche – basierend auf dem Linux-Kernel, den GNU-Tools, graphischen Benutzeroberflächen und zahlreichen weiteren Applikationen – komplette Softwaresysteme zusammenstellen. Diese, mit unterschiedlichen Zusatzleistungen, wie z.B. einem Installationsprogramm und einem Handbuch versehenen *Distributionen* werden i.d.R. auf CD-ROM zum Kauf angeboten. Der Preis, der für eine solche Distribution zu entrichten ist, bezieht sich nicht auf die Software selbst – welche im Falle von OSS ohnehin frei von Lizenzgebühren zur Verfügung steht – sondern vielmehr auf die von den Distributoren übernommene Integration zahlreicher Softwarepakete zu einem kompletten System sowie auf ihre zusätzlichen spezifischen Leistungen<sup>47</sup>. Zu den bekanntesten Distributoren zählen Red Hat, Caldera, Debian, FreeBSD, Mandrake und SuSE<sup>48</sup>.

<sup>44</sup> Vgl. Köppen/Nüttgens (2000), S. 237 - 241.

<sup>45</sup> Eine der zentralen Entwicklungen von Cygnus Solutions ist das GNUPro Development Kit. Hierbei handelt es sich um eine freie, integrierte Entwicklungsumgebung (IDE), die u.a. unter der GPL veröffentlichte Compiler, Debugger und Linker umfasst.

<sup>46</sup> Vgl. Tiemann (1999).

<sup>47</sup> Vgl. Grassmuck (2001b) und Lerner/Tirole (2001), S. 820.

<sup>48</sup> Führende Distributoren tendieren in jüngster Zeit dazu, neben ihren Distributionen Accessories, technischen Support oder Schulungen anzubieten.

## Schnittstelle zwischen kommerziellen Unternehmen und der Open Source-Community

Gegenstand dieses Konzepts ist eine gebührenpflichtige Vermittlerfunktion zwischen Unternehmen, die ihre bislang proprietäre Software einer OSS-Lizenz unterstellen wollen, und der Open Source-Community. Derartige Vermittler können neben der eigentlichen DV-technischen Unterstützung<sup>49</sup> des Projekts – die Akzeptanz der Community vorausgesetzt – dessen Leitung übernehmen, um möglichen Vorbehalten der Open Source-Community gegenüber kommerziellen Softwareunternehmen entgegenzuwirken<sup>50</sup>. Darüber hinaus sind in diesem Zusammenhang beratende Dienstleistungen, z.B. hinsichtlich des erfolgreichen Aufbaus einer projektspezifischen Community, denkbar.

Collab.Net bspw. bietet mit SourceCast und VA Linux Systems mit der SourceForge Portal Edition eine solche Schnittstelle zwischen kommerziellen Unternehmen und der Community an<sup>51</sup>.

Den Beteiligten<sup>52</sup> stehen hierbei verschiedene Werkzeuge und Dienste für eine gemeinsame, weltweit über das Internet verteilte Softwareentwicklung zur Verfügung. Außenstehenden Teilnehmern können ggf. verschiedenartige Möglichkeiten der Beteiligung geboten sowie beliebige Zugriffsrechte eingeräumt werden. Mit zusätzlichen Entwicklern und aufgrund deren unentgeltlicher Teilnahme kann ein Unternehmen möglicherweise den Software-Entwicklungsprozess beschleunigen und seine Entwicklungskosten reduzieren<sup>53</sup>.

---

<sup>49</sup> Vgl. Abschnitt 3.3.

<sup>50</sup> Vgl. Lerner/Tirole (2001), S. 825.

<sup>51</sup> Vgl. Collab.Net (Hrsg.) (2001) und VA Linux Systems (Hrsg.) (2001b). Zu den Kunden von Collab.Net, dessen Chief Technical Officer Brian Behlendorf vom Apache-Projekt ist, zählen u.a. Hewlett-Packard, Sun Microsystems und Oracle.

<sup>52</sup> Neben Entwicklern aus der Open Source-Community können sich bspw. die Kunden des kommerziellen Softwareunternehmens an einem Projekt beteiligen.

<sup>53</sup> In Deutschland hat sich das auf der SourceForge.net-Software basierende BerliOS-Projekt des Forschungsinstituts für offene Kommunikationssysteme (FOKUS) zum Ziel gesetzt, "die unterschiedlichen Interessengruppen im Umfeld der Open-Source-Software (OSS) zu unterstützen und dabei eine Vermittlerfunktion anzubieten." Entwicklern steht auch hier eine zentrale DV-technische Infrastruktur für die Koordination einer verteilten Softwareentwicklung zur Verfügung. Um darüber hinaus das Angebot an freier Software zu erweitern, können Interessierte in der *Open-Source Software-Börse* Beträge für die Entwicklung einer benötigten, jedoch noch nicht existierenden OSS-Lösung ausschreiben. Vgl. BerliOS (Hrsg.) (2001). Eric S. Raymond spricht in diesem Zusammenhang von einem "reverse-auction model to funding open-source development." Raymond (2001g), S. 160. Die Free Software Foundation gründete mit *Savannah* ein ähnliches Projekt für die Entwicklung freier Software.



## 4.3 Möglichkeiten der Beteiligung an Open Source Software-Projekten

Aufgrund der Tatsache, dass freie Softwareprojekte weder über Kapitaleinlagen verfügen noch einen Eigentümer im rechtlichen Sinne haben, ist eine *Übernahme* derartiger Projekte durch finanzielle Maßnahmen von Personen oder Unternehmen nicht möglich. Darüber hinaus liegt der Community viel an ihrer Offenheit und Unabhängigkeit, weshalb anderweitige materielle Angebote mit dem Versuch der Einflussnahme i.d.R. erfolglos bleiben<sup>54</sup>.

Da kapitalmäßige Beteiligungstransaktionen als traditionelle Maßnahmen unternehmenspolitischer Zielerreichung<sup>55</sup> demnach auszuschließen sind, bleibt einem Unternehmen nur der Versuch, über personelle Beziehungen zur Open Source-Community an deren Entwicklungen teilzuhaben und ggf. von ihnen zu profitieren.

Hierbei lassen sich zwei ähnliche Formen der Unterstützung von bzw. der Beteiligung an OSS-Projekten beobachten. Einerseits zeigt sich, dass Unternehmen einzeln Kernentwickler verschiedener Projekte einstellen. Andererseits erlauben sie teilweise ihren Angestellten, in geeigneten Projekten mitzuwirken. Die erstgenannte Form der Beteiligung ist in der Praxis häufiger zu beobachten. Neben dem bereits erwähnten Verlag O'Reilly & Associates sind es vor allem Distributoren, die führende Entwickler unterschiedlicher Projekte in Forschungs- und Entwicklungsabteilungen, wie z.B. in den SuSE Linux Labs und den Red Hat Advanced Development Labs, beschäftigen. Beispiele für die zweite Möglichkeit der Beteiligung finden sich in VA Linux Systems, Sendmail und IBM, welches eigens Personal für das Open Source Development Lab sowie das Linux Standard Base-Projekt abstellt<sup>56</sup>.

Kommerzielle Unternehmen können mit derartigen Beteiligungen unterschiedliche Ziele verfolgen. Zunächst bietet sich ihnen die Möglichkeit, über allgemeine technische Entwicklungen in den entsprechenden Bereichen besser informiert zu sein. Daneben können sie die Absichten verfolgen, für sinnvoll erachtete Neuerungen aus dem freien Softwareprojekt schneller in ihre eigenen Produkte zu integrieren oder talentierte Entwickler aus der Community anzuwerben und einzustellen<sup>57</sup>. Weiterhin können Unternehmen auf diese Weise mehr über die Stärken und Schwächen des OSS-Modells

<sup>54</sup> Vgl. Glasl (2001). Ziel einer Einflussnahme kann bspw. die Änderung der Lizenzbestimmungen sein. Einige Unternehmen aus dem OSS-Umfeld, wie z.B. Collab.Net und Cobalt Networks, wurden jedoch erst aufgrund finanzieller Unterstützung durch Venture Capital-Unternehmen gegründet bzw. vor der Insolvenz bewahrt.

<sup>55</sup> Vgl. Schierenbeck (1973), S. 17.

<sup>56</sup> Vgl. Köppen/Nüttgens (2000), S. 232, Lerner/Tirole (2000), S. 20 und BMWi (Hrsg.) (2001), S. 10 f.

<sup>57</sup> Aufgrund der Tatsache, dass die Beiträge einzelner Teilnehmer festgehalten und veröffentlicht werden, können OSS-Entwickler ihre Kenntnisse und Fähigkeiten weltweit unter Beweis stellen, vgl. Lerner/Tirole (2000), S. 16 ff.

erfahren und ggf. ihre internen Abläufe und Strukturen daran anpassen. Distributoren und Unternehmen wie O'Reilly & Associates, für die freie Software die wesentliche Geschäftsgrundlage darstellt, werden schließlich darum bemüht sein, durch die genannten Beteiligungen ihr Ansehen innerhalb der Open Source-Community zu festigen und zu verbessern sowie eine weitere Verbreitung von OSS zu fördern<sup>58</sup>.

Derartige Anstrengungen sind offensichtlich keineswegs uneigennützig. Es ist vielmehr anzunehmen, dass mit o.g. Beteiligungen der Versuch unternommen werden soll, einen Wettbewerbsvorteil gegenüber konkurrierenden Unternehmen zu erreichen. Scheinbar gelangten verschiedene Unternehmen zu der Erkenntnis, dass durch derartige Maßnahmen indirekt Erlöse erzielbar sind, die langfristig die Kosten der Beteiligungen decken oder übersteigen können<sup>59</sup>.

## 4.4 Übertragbarkeit organisatorischer Aspekte des Open Source-Modells auf Unternehmen

Obwohl die Gegebenheiten des Open Source-Modells mit den Bedingungen und Zielsetzungen ökonomischen Handelns nur selten vergleichbar sind, können sich dennoch neue Formen der Organisation und Zusammenarbeit sowohl innerhalb eines Unternehmens als auch unternehmensübergreifend eröffnen. Offensichtlich kann ein freies Softwareprojekt als eine dezentrale Organisation durch eine Abkehr von formal-hierarchischen Strukturen und eine Anwendung kooperativer Entscheidungsprozesse auch in großem Umfang weltweit verteilt über ein Netzwerk funktionieren. Diese Erkenntnis ist die Grundlage der folgenden Überlegungen.

### 4.4.1 Unternehmensübergreifende Netzwerke

Zu den wesentlichen Merkmalen der Open Source-Community gehört die offene und kooperative Arbeit an einem gemeinsamen Ziel. Durch arbeitsteilige Maßnahmen können sich individuell unterschiedlich begabte Teilnehmer auf eine bestimmte Tätigkeit konzentrieren und auf diese Weise ihren größtmöglichen Beitrag zur Zielerreichung leisten.

Die Übertragung dieses Prinzips auf das unternehmerische Handeln kann die Bildung von Unternehmensnetzwerken begünstigen. Derartige Netzwerke sind dadurch gekennzeichnet, dass formal unabhängige und rechtlich selbständige Unternehmen auf eine Weise kooperieren, die über rein ökonomisch motivierte Beziehungen deutlich hinausgeht. In Analogie zur Open Source-Community kann sich hierbei jedes

<sup>58</sup> Vgl. Lerner/Tirole (2001), S. 824 und Lerner/Tirole (2000), S. 26.

<sup>59</sup> Vgl. Lerner/Tirole (2000), S. 16 f. und Weber (2001).

Netzwerkmitglied auf diejenige Leistung beschränken, für die es die größte unternehmensspezifische Kompetenz besitzt. Eine solche Spezialisierung wiederum kann vor allem im Bereich der Forschung und Entwicklung zu Kostenvorteilen führen<sup>60</sup>.

Neben der Möglichkeit eines gemeinschaftlichen Lernprozesses und den genannten Kostenvorteilen kann das angestrebte Ziel eines solchen Unternehmensnetzwerks darin bestehen, durch koordinierte Strategien und kooperative Verhaltensweisen über eine kollektive Effizienzsteigerung die individuellen Wettbewerbspositionen zu verbessern<sup>61</sup>.

Voraussetzung hierfür ist jedoch einerseits die Bereitschaft, von der Realisation eigener Vorteile auf Kosten der übrigen Netzwerkmitglieder abzusehen sowie andererseits die Bereitstellung sämtlicher erfolgsrelevanten Informationen. Mögliche Vorteile aus Unternehmensnetzwerken sind demnach wesentlich davon abhängig, inwieweit die beteiligten Unternehmen gegenseitiges Vertrauen aufbauen können<sup>62</sup>. Schließlich ist für den Fortbestand eines solchen Netzwerks entscheidend, dass seine Mitglieder die möglichen Vorteile realisieren und somit einer Auflösung des Netzwerks entgegengewirkt werden kann<sup>63</sup>.

#### 4.4.2 Unternehmensinterne Arbeitsorganisation

Nur locker zusammenhängende Organisationseinheiten zeichnen sich im Gegensatz zu fest miteinander verbundenen Elementen durch eine geringere Abhängigkeit von vor- und nachgelagerten Prozessen aus und ermöglichen flexiblere Reaktionen auf unvorhergesehene Veränderungen. Störungen in derartigen Organisationseinheiten bleiben zunächst auf diese begrenzt, ein Übergriff auf das gesamte System erfolgt weniger schnell und kann somit leichter verhindert werden<sup>64</sup>.

Darüber hinaus lässt sich in der organisatorischen Praxis beobachten, dass redundante, parallele Strukturen in Konkurrenz zueinander um knappe Ressourcen treten, wodurch der ineffizienten Nutzung begrenzter Ressourcen entgegengewirkt werden kann<sup>65</sup>. Mit einer redundanten, parallelen Organisation ist es zudem möglich, ein zuverlässiges System zu gestalten, welches aus tendenziell unzuverlässigen Elementen besteht<sup>66</sup>.

Aufgrund dieser Beobachtungen kann die Übertragung der vergleichsweise locke-

<sup>60</sup> Vgl. Nüttgens/Tesei (2001c), S. 13 f. und S. 18.

<sup>61</sup> Vgl. Nüttgens/Tesei (2001c), S. 13.

<sup>62</sup> Vgl. Nüttgens/Tesei (2001c), S. 15 f.

<sup>63</sup> In ähnlicher Weise ist die kontinuierliche Unterstützung eines freien Softwareprojekts durch die Community wesentlich von einem erkennbaren Projektfortschritt abhängig.

<sup>64</sup> Vgl. Nüttgens/Tesei (2001b), S. 11 f.

<sup>65</sup> Vgl. Behlendorf (1999), S. 162. Im Open Source-Umfeld zeigt sich in diesem Zusammenhang, dass Projekte mit ähnlichen Zielsetzungen um Unterstützung durch Entwickler aus der Community konkurrieren.

<sup>66</sup> Vgl. Nüttgens/Tesei (2001b), S. 10 f.

ren, nur wenig koordinierten Form der Zusammenarbeit sowie die teilweise mehrfache Besetzung und Ausführung wichtiger Funktionen innerhalb einer Open Source-Community die Entstehung neuer Formen der Arbeitsorganisation fördern. Denkbar wäre hier bspw. die Bildung teilautonomer Arbeitsgruppen oder Abteilungen sowie mehrerer austauschbarer Teams, welche gleiche oder ähnliche Aufgaben erfüllen. Neben den genannten Konsequenzen können derartige Maßnahmen aufgrund eines Wettbewerbs zwischen den Gruppen ggf. zu einer fristgerechten Aufgabenerfüllung oder zu qualitativ höherwertigen Ergebnissen beitragen<sup>67</sup>.

---

<sup>67</sup> Vgl. Nüttgens/Tesei (2001b), S. 8 f.

# Kapitel 5

## Generalisierung

Daten<sup>1</sup> werden im Internet zur flüchtigen Ware. Zeitschriften, Bücher, Musiktitel und Videos sind – z.T. entgegen einer Erlaubnis des Inhabers der Urheberrechte und somit illegal – digitalisiert verfügbar und nahezu uneingeschränkt reproduzierbar. In Netzwerken zirkulieren inzwischen Daten, deren Übermittlung lange Zeit Telefon, Hörfunk, Fernsehen oder Printmedien vorbehalten waren. Die wesentlichen technischen Voraussetzungen für die Übertragung des Open Source-Modells auf die Entstehung, Entwicklung und Verbreitung digitaler Informationen scheinen offensichtlich gegeben zu sein<sup>2</sup>.

Obwohl sich mittels elektronischer Datenverarbeitung prinzipiell jegliche digitalen Informationen manipulieren und vervielfältigen lassen können, bestehen jedoch Unterschiede in der individuellen Nutzung und Bedeutung dieser Informationen. Aus diesem Grund gilt es zunächst zu überlegen, ob sich die verschiedenartigen Daten überhaupt für eine dezentrale, kooperative Entwicklung eignen<sup>3</sup>.

Im Gegensatz zu solchen Informationen, die vorwiegend im Bereich der Bildung und Wissenschaft entstehen und genutzt werden, erscheint die gemeinschaftliche Weiterentwicklung derjenigen Daten, welche in erster Linie der *Würdigung* dienen sollen, sich dem Ausdruck der individuellen Kreativität des Urhebers zuschreiben lassen und nur in geringem Maße formalisierbar sind – bei denen es sich also im weitesten Sinne

---

<sup>1</sup> Unter *Daten* sollen Zeichen oder kontinuierliche Funktionen verstanden werden, die aufgrund von bekannten oder unterstellten Abmachungen und zum Zwecke der elektronischen Verarbeitung Informationen darstellen. Ihre Erscheinungsform kann schriftlich, akustisch und/oder bildlich sein. Vgl. Biethahn/Mucksch/Ruf (1996), S. 4 f. und Mertens/Bodendorf/König/Picot/Schumann (1996), S. 54 f. Die Begriffe *Daten* und *Information* werden im folgenden synonym zueinander genutzt.

<sup>2</sup> Vgl. Köppen/Nüttgens (2000), S. 231 und Meyer (1998), S. 178.

<sup>3</sup> Darüber hinaus müsste im einzelnen geklärt werden, worin der *Quellcode* der Daten besteht.

um Unterhaltung und/oder Kunst handelt – weniger sinnvoll<sup>4</sup>. Unter der Annahme, dass akustische und bildliche Daten sowie Kombinationen dieser Erscheinungsformen in erster Linie der Unterhaltung dienen bzw. der Kunst zuzuschreiben sind und dass ihr Urheber – aufgrund von Bedenken, dass ihr ursprüngliches Motiv bzw. ihr Inhalt verlorengehen oder entfremdet werden – einer gemeinschaftlichen, nicht vollständig kontrollierbaren Manipulation dieser Informationen ohnehin in den wenigsten Fällen zustimmen wird, sollen derartige Daten im folgenden nicht weiter berücksichtigt werden<sup>5</sup>. Zudem scheint vor allem die organisatorische Umsetzung einer nicht zuletzt geographisch verteilten Entwicklung derartiger Informationen nur schwer realisierbar zu sein. Aus diesen Gründen konzentrieren sich die folgenden Überlegungen auf schriftliche Daten<sup>6</sup>.

Nachfolgend soll untersucht werden, inwiefern das Konzept einer verteilten, DV-technisch unterstützten und lizenzrechtlichen Bestimmungen unterliegenden Softwareentwicklung auf Texte im Allgemeinen angewendet werden kann.

## 5.1 Technische Übertragbarkeit

Es ist anzunehmen, dass die Nutzung der vorhandenen, für freie Softwareprojekte typischen DV-technischen Mittel<sup>7</sup> für eine verteilte, gemeinschaftliche Fortschreibung von schriftlichen Daten jeglicher Art ohne weiteres möglich ist. Voraussetzung ist lediglich, dass der Zeichenvorrat, der die Grundlage dieser Daten bildet, von den einzusetzenden DV-Werkzeugen unterstützt wird. Gleichmaßen kann das Concurrent Versions System (CVS) mit beliebigen Textdateien umgehen und ist prinzipiell für einen solchen Zweck nutzbar.

<sup>4</sup> Vgl. Stallman (2001b). Es gibt bereits Projekte, die der sog. *Net Art* zuzuschreiben sind. Obwohl jedoch Net Art ebenfalls auf Software basiert, haben derartige Projekte mit dem OSS-Konzept lediglich das Internet als Kommunikations- und Publikationsmedium gemeinsam. Vgl. Shulgin (2001). Darüber hinaus gibt es seit kurzem mit den Varianten der OpenMusic License und der EFF Open Audio License verschiedene Lizenzen, welche im wesentlichen versuchen, den Gedanken freier Software auf akustische und audiovisuelle Daten zu übertragen. Die Design Science License schließlich wurde als eine das Copyleft-Konzept umsetzende Lizenz für beliebige, dem Urheberrecht unterstehenden Werke formuliert. Zum gegenwärtigen Zeitpunkt ist jedoch nicht absehbar, welche Bedeutung ihnen zukommen wird. Vgl. OpenMusic Initiative (Hrsg.) (2001), Electronic Frontier Foundation (Hrsg.) (2001) sowie Stutz (2001).

<sup>5</sup> In diesem Zusammenhang ist ferner anzunehmen, dass die Vielzahl der Audio- und Video-Dateiformate eine Übertragung des OSS-Konzepts auf akustische und bildliche Daten erschweren kann. Trotz verschiedener Kompressionsverfahren verursacht schließlich die Bearbeitung, Speicherung und Übermittlung von Audio- und Videodateien einen vergleichsweise hohen rechtechnischen Aufwand.

<sup>6</sup> Der Quellcode einer Software, bei dem es sich um *lesbare* Anweisungen, formuliert in einer Programmiersprache handelt, gehört demnach ebenfalls zu den schriftlichen Daten, welche allgemein aus Buchstaben, Ziffern und Sonderzeichen eines gegebenen endlichen Zeichenvorrats bestehen und aufgrund bestimmter Regeln zusammengesetzt werden.

<sup>7</sup> Vgl. Abschnitt 3.3.

## 5.2 Organisatorische Übertragbarkeit

Eine wesentliche Eigenschaft der Open Source-Community ist die ständige Überprüfung und Kritik des Quellcodes durch die Projektteilnehmer mit dem Ziel der Fehlerbehebung. Ein solcher nicht-linearer Peer Review-Prozess ist prinzipiell für schriftliche Daten im Allgemeinen denkbar. Es ist jedoch fraglich, ob Texte generell sinnvoll auf diese Weise fortgeschrieben oder aktualisiert werden können, da im Gegensatz zu Software die *Korrektheit* zahlreicher Texte häufig nicht überprüft werden kann und subjektiven Vorstellungen unterliegt<sup>8</sup>. Davon abgesehen ist jedoch anzunehmen, dass die Übertragung der in Abschnitt 3.2 genannten organisatorischen Aspekte freier Softwareprojekte auf beliebige textbasierte Projekte möglich ist<sup>9</sup>.

## 5.3 Rechtliche Übertragbarkeit

In der Vergangenheit wurden eine Vielzahl von Lizenzen mit dem Ziel formuliert, den Gedanken freier Software auf beliebige Texte zu übertragen. Die Initiatoren dieser Lizenzen versuchten i.d.R., die Bestimmungen der GPL an derartige Daten anzupassen bzw. die GPL zu verallgemeinern. Da ihr Inhalt überwiegend das Urheberrecht betrifft<sup>10</sup>, soll zunächst die grundsätzliche Verträglichkeit der einer Vielzahl derartiger Lizenzen zugrundeliegenden GPL mit dem deutschen Urheberrechtsgesetz (UrhG) untersucht werden<sup>11</sup>, bevor einige dieser sog. *Open Content*-Lizenzen vorgestellt werden.

### 5.3.1 Exkurs: Konformität der GPL mit dem deutschen Urheberrechtsgesetz

Zu den zentralen Bestimmungen der GPL zählt die uneingeschränkte Erlaubnis, jederzeit Vervielfältigungen des *Werkes*<sup>12</sup> vornehmen und es auf beliebigen Medien verbreiten zu dürfen. Nach §§ 15, 29 UrhG steht dem Urheber dagegen zunächst das alleinige und unübertragbare Recht an der Verwertung seines Werkes zu. Die Formulierungen der Ziffern 4 und 5 GPL deuten jedoch darauf hin, dass durch das

<sup>8</sup> Um eine Verifizierung beliebiger Texte überhaupt durchführen zu können, müsste zunächst definiert werden, worin ihre Korrektheit besteht.

<sup>9</sup> Einen ersten Ansatz hierzu bildet das Projekt Open Theory, vgl. Open Theory (Hrsg.) (2001).

<sup>10</sup> Aus diesem Grund soll an dieser Stelle auf die Untersuchung der Verträglichkeit des in den meisten (Software-)Lizenzen enthaltenen Gewährleistungs- und Haftungsausschlusses mit dem Gesetz zur Regelung des Rechts der Allgemeinen Geschäftsbedingungen (AGBG) verzichtet werden.

<sup>11</sup> Obwohl die GPL in den USA formuliert und auf die dortigen Verhältnisse ausgerichtet wurde, ist nach dem Territorialitätsprinzip des internationalen Urheberrechts davon auszugehen, dass bei Fragen, die die Inhalte und den Schutz von Urheberrechten betreffen, das Recht des jeweiligen Schutzlandes anzuwenden ist. Vgl. Metzger/Jaeger (2001) S. 58 f. und Schulze (2001), S. 277 f.

<sup>12</sup> Nach dem UrhG wird jede, den urheberrechtlichen Bestimmungen unterliegende Schöpfung als *Werk* bezeichnet.

Zustandekommen des Lizenzvertrags jedem Lizenznehmer lediglich ein einfaches und an verschiedene auflösende Bedingungen<sup>13</sup> geknüpftes Nutzungsrecht an dem Werk gemäß § 31 UrhG Absatz 1, 2 eingeräumt wird<sup>14</sup>. Durch diese Bestimmungen ist gewährleistet, dass das Urheberrecht bei dem ursprünglichen Autor der Software verbleibt.

Problematisch ist dagegen die in Ziffer 2 GPL eingeräumte Erlaubnis, beliebige Bearbeitungen des Werkes herzustellen und zu verbreiten. Dem Urheber steht in diesem Zusammenhang nach § 14 UrhG das Recht zu, Entstellungen und Beeinträchtigungen, die seine berechtigten geistigen oder persönlichen Interessen am Werk gefährden können, zu untersagen<sup>15</sup>. Für den Lizenznehmer besteht demnach prinzipiell das Risiko, dass der Urheber das eingeräumte Nutzungsrecht nach § 14 UrhG widerruft.

Ein weiteres Problem wird durch Ziffer 9 GPL hervorgerufen. Dieser Punkt besagt, dass dem Urheber die Möglichkeit zusteht, durch einen entsprechenden Vermerk sein Werk der jeweils aktuellen Version der GPL, welche bei Bedarf an geänderte Bedingungen – bspw. aufgrund technologischen Fortschritts – angepasst werden kann, zu unterstellen<sup>16</sup>. Durch einen solchen Vermerk können somit zukünftig mögliche Verwendungsmöglichkeiten des Werkes lizenziert werden. § 31 Absatz 4 UrhG erklärt jedoch die Einräumung von Nutzungsrechten für noch nicht bekannte Nutzungsarten für ungültig, ein solcher Vermerk ist demnach unwirksam<sup>17</sup>.

Zusammenfassend kann festgehalten werden, dass die GPL grundsätzlich mit dem deutschen Urheberrecht konform ist und ihre zentralen Bestimmungen überwiegend als rechtlich bindend bewertet werden. Dennoch sind nach diesem Gesetz einige ihrer Klauseln unwirksam. Die Free Software Foundation (FSF) bemüht sich nicht zuletzt aus diesem Grund darum, spezielle Versionen der GPL zu erstellen, welche an die Bedürfnisse und Besonderheiten eines Staates und dessen gesetzliche Bestimmungen angepasst sind<sup>18</sup>.

### 5.3.2 Open Content-Lizenzen

Zu den nach dem UrhG geschützten Werken der Literatur, Wissenschaft und Kunst gehören u.a. Sprachwerke, wie bspw. Schriftwerke, Reden und Computerprogramme. Voraussetzung für die Anwendbarkeit des UrhG ist lediglich, dass es sich bei diesen

---

<sup>13</sup> Vgl. Ziffer 4 GPL.

<sup>14</sup> Vgl. Schulze (2001), S. 73 und S. 156 ff.

<sup>15</sup> Vgl. Schulze (2001), S. 84 f.

<sup>16</sup> In Ziffer 9 GPL heißt es u.a.: “If the Program specifies a version number of this License which applies to it and “any later version”, you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation.”

<sup>17</sup> Vgl. Metzger/Jaeger (2001), S. 67 f. und Schulze (2001), S. 166 ff.

<sup>18</sup> Vgl. Free Software Foundation (Hrsg.) (2001g).



Werken um “persönliche geistige Schöpfungen” nach § 2 UrhG handelt.

Im Open Source-Umfeld finden sich verschiedene Lizenzen, deren Autoren im wesentlichen das Ziel verfolgten, den Gedanken freier Software auf beliebige schutzfähige Schriftstücke zu übertragen<sup>19</sup>. Derartige Lizenzen, welche größtenteils in Anlehnung an die GPL formuliert wurden, erlauben i.d.R. jedem, die ihr unterliegenden Dokumente – ggf. bis auf sog. *unveränderliche Abschnitte*<sup>20</sup> – uneingeschränkt zu nutzen, zu modifizieren, zu kopieren und zu verteilen, wobei modifizierte Dokumente wiederum derselben Lizenz zu unterliegen haben. Darüber hinaus sind das Datum, der Autor und der Inhalt der Veränderungen an geeigneter Stelle kenntlich zu machen.

Zu den gebräuchlichsten dieser Open Content-Lizenzen zählen die GNU Free Documentation License der Free Software Foundation<sup>21</sup>, die OpenContent License<sup>22</sup> sowie die Open Publication License<sup>23</sup>.

Die rechtlichen Rahmenbedingungen für eine offene und verteilte Entwicklung bzw. Fortschreibung beliebiger schriftlicher Daten nach dem Open Source-Modell sind demnach gegeben.

## 5.4 Fazit

Die Übertragung des OSS-Modells auf beliebige Texte scheint prinzipiell möglich zu sein, einige vorwiegend softwarebezogene Dokumentationen wurden zudem bereits unter entsprechenden Lizenzen veröffentlicht<sup>24</sup>. *Open Source*-Projekte, welche *herkömmliche* Texte außerhalb des softwaretechnischen Bereichs zum Gegenstand haben, sind jedoch lediglich ansatzweise auszumachen. Eine mögliche Ursache hierfür könnte darin bestehen, dass die Anzahl potenzieller Interessenten für die Fortschreibung derartiger Texte nicht groß genug ist. Zudem scheint der mit der Aufbereitung

<sup>19</sup> Indem die GPL den *Source Code* lediglich als “the preferred form of the work for making modifications to it” definiert, ist sie gleichermaßen auf beliebige Daten anwendbar. Voraussetzung ist, dass ein solcher *Source Code* im Einzelfall existiert und bestimmt werden kann.

<sup>20</sup> Vgl. Anhang A: GNU Free Documentation License.

<sup>21</sup> Die GNU Free Documentation License unterscheidet nach der Art und Weise, in der die ihr unterliegenden Dokumente vervielfältigt und veröffentlicht werden, transparente (*transparent*) von nicht-transparenten (*opaque*) Kopien. Eine transparente Kopie ist dadurch gekennzeichnet, dass sie in einem allgemein zugänglichen Format mit einfachen Texteditoren eingesehen und verändert werden kann. Bei der Veröffentlichung von Dokumenten in einem nicht-transparenten, d.h. nicht in dieser Form lesbaren und i.d.R. proprietären Format, muss jedoch der Zugriff auf die transparente Version gewährleistet sein.

<sup>22</sup> *Content* wird hier lediglich als “anything that isn’t just executable” verstanden. Vgl. OpenContent (Hrsg.) (2001a).

<sup>23</sup> Die Open Publication License bietet dem Autor zudem die Optionen, ohne seine vorherige Erlaubnis weder wesentliche Änderungen an seinem Dokument durchführen zu dürfen noch sein Dokument oder von diesem abgeleitete Arbeiten in “standard (paper) book”-Form zu verbreiten. Nimmt der Autor nur eine dieser beiden Optionen wahr, handelt es sich nicht mehr um ein *freies* Dokument. Vgl. OpenContent (Hrsg.) (2001b) und (2001c).

<sup>24</sup> Vgl. Linux Documentation Project (Hrsg.) (2001).

der Texte für eine Veröffentlichung im Internet verbundene Aufwand vergleichsweise hoch zu sein. Schließlich ist anzunehmen, dass eine wesentliche Schwierigkeit bei der Übertragung des OSS-Modells auf beliebige Schriftstücke in einer geeigneten und sinnvollen Modularisierung derselben besteht.

Eine Generalisierung des Open Source-Modells erscheint theoretisch möglich. Jedoch ist zum gegenwärtigen Zeitpunkt – nicht zuletzt aus den eingangs genannten Gründen – nicht abzusehen, dass entsprechende Projekte jenseits der Softwareentwicklung die Popularität freier Softwareprojekte erlangen werden.

# Kapitel 6

## Fazit und Ausblick

Open Source Software konnte in vielen Bereichen der Softwarenutzung zweifelsohne an Bedeutung gewinnen und sich gegenüber proprietärer Software durchsetzen, der Prozess ihrer Entwicklung und Verbreitung scheint sich trotz vereinzelter Anstrengungen traditioneller, kommerziell orientierter Unternehmen nicht aufhalten zu lassen. Die Leistungen einer vermeintlich unprofessionellen und unwissenschaftlichen Hackerkultur scheinen zudem Kritiker, welche die Konzepte und Methoden der Open Source-Community mit “Graswurzeltätigkeiten<sup>1</sup>” vergleichen, eines Besseren zu belehren. Freie Software wird proprietäre Software nicht völlig ersetzen oder verdrängen, sie kann jedoch den Softwaremarkt zunehmend beeinflussen und verändern.

Vertreter der Wirtschaft äußern in diesem Zusammenhang die Ansicht, freie Software gefährde Arbeitsplätze und behindere die Realisierung innovativer Entwicklungen<sup>2</sup>. Dagegen scheinen jedoch gerade die Bereiche Internet und Kommunikation, in denen offene Standards von wesentlicher Bedeutung sind, Perspektiven für Unternehmen im Open Source-Umfeld zu eröffnen<sup>3</sup>. Überschaubare softwarebezogene Investitionsausgaben können zudem die Gründung derartiger Unternehmen begünstigen. Darüber hinaus ist anzunehmen, dass die wirtschaftliche Bedeutung freier Software im Allgemeinen wesentlich von einem zunehmenden Angebot qualitativ hochwertiger Produkte abhängen wird. Letztendlich wird in erster Linie der Wettbewerb als selektiver Mechanismus darüber entscheiden, inwiefern sich das Modell freier Software in ökonomischer Hinsicht durchsetzen kann<sup>4</sup>.

---

<sup>1</sup> Endres (2000), S. 317.

<sup>2</sup> Vgl. bspw. Mundie (2001). “Wenn man öffentliche Mittel dafür einsetzt, um nichtkommerzielle Gebrauchsoftware zu verbreiten, macht man nichts anderes, als einer Industrie, die Arbeitsplätze schafft, das Wasser abzugraben.” Endres (2000), S. 318. Anstelle auf ihre vermeintliche Benachteiligung hinzuweisen könnte die Softwareindustrie überlegen, inwiefern sie das Potenzial der OSS-Entwickler für ihre Zwecke nutzen kann.

<sup>3</sup> Die Bedeutung dieser beiden Bereiche unterstreicht die Tatsache, dass auf sie ca. 30% der über SourceForge.net koordinierten Projekte entfallen, vgl. VA Linux Systems (2001a).

<sup>4</sup> Inwieweit die Bedingungen wirtschaftlichen Handelns, welche auf einem Missverhältnis von knappen Gütern zu unbegrenzten Bedürfnissen basieren, überhaupt auf OSS als freies, öffentliches Gut zutreffen, bleibt gleichermaßen abzuwarten.

Möglicherweise scheint die eigentliche Innovation des Open Source-Modells weniger in der Software selbst, sondern vielmehr in der Reflektion eines organisatorischen Trends zur zunehmenden, grenzenüberschreitenden Kooperation von Menschen über Netzwerke zu bestehen. Die Effizienz freier Softwareprojekte beruht vor allem darauf, unterschiedliche Individuen und Kompetenzen zu integrieren und deren Wissen zu verwerten. Die Open Source-Community gelangte zudem offensichtlich zu der Erkenntnis, dass das Netzwerk nicht nur ihr Kommunikationsmedium, sondern gleichermaßen ihr Publikationsmedium ist. In diesem Zusammenhang besteht jedoch die Gefahr, dass durch ihre zunehmende Popularität und der für jeden offen stehenden Beteiligung Umstände herbei geführt werden, welche sich nachteilig auf die Entwicklung freier Software auswirken können<sup>5</sup>. Der Zwang zum Konsens und die Notwendigkeit zahlreicher Kompromisse aufgrund organisatorischer Skalierungsprobleme können das gemeinschaftliche Streben nach der bestmöglichen Lösung unterwandern und Auflösungserscheinungen begünstigen. Ein Ausweg aus diesem Dilemma bestünde in der Softwareentwicklung unter weitgehendem Ausschluss der Öffentlichkeit, jedoch würde sich die Open Source-Community hiermit den exklusiven Charakter proprietärer Softwareentwicklung, gegen den sie einst angetreten ist, zu eigen machen<sup>6</sup>.

Open Source Software kombiniert Erfahrungen aus der Entwicklung und dem Vertrieb von Software mit den Bedingungen einer zunehmenden, weltweiten Vernetzung. Untersuchungen über die Prämissen und Konzepte eines derartigen offenen Systems, das auf der eigenmotivierten Bereitschaft seiner Mitglieder basiert, Informationen uneingeschränkt mit der Gemeinschaft auszutauschen, können zukünftig von allgemeinem Interesse sein. Die Fähigkeit, die mit den Konzepten der Open Source-Community verbundenen Fragen zu beantworten, kann im gleichen Maße steigen, wie sich die Open Source-Bewegung selbst weiterentwickelt. Freie Software vereint technische und gesellschaftliche Entwicklungen unserer Zeit. Inwiefern sie sich als ein neues Modell für die Softwareentwicklung im Speziellen und für eine offene, dezentralisierte Kooperation im Allgemeinen eignet, wird sich in Zukunft herausstellen<sup>7</sup>.

---

<sup>5</sup> Vgl. Bezroukov (2001) und Zawinski (2001).

<sup>6</sup> Vgl. Hofmann (2001).

<sup>7</sup> Vgl. Nüttgens/Tesei (2001c), S. 21 f.



# Literatur- und Quellenverzeichnis

Aladdin Enterprises (Hrsg.) (2001): Aladdin Free Public License,  
<http://www.cs.wisc.edu/~ghost/doc/cvs/Public.htm>, 2000-09-18, Abruf am  
2001-09-28. (Version 9).

Apple Computer (Hrsg.) (2001): Darwin,  
<http://www.opensource.apple.com/projects/darwin/>, Abruf am 2001-09-21.

Balzert, H. (2001): Lehrbuch der Software-Technik, Bd. 1. Software-Entwicklung,  
2. Aufl., Heidelberg u.a.

Bayer, M. (2001): Stehen die Linux-Geschäftsmodelle vor dem Aus?,  
<http://www1.computerwoche.de/index.cfm?pageid=254&artid=24495>, 2001-09-21,  
Abruf am 2001-10-10.

Behlendorf, B. (1999): Open Source as a Business Strategy.  
In: DiBona, C./Ockman, S./Stone, M. (1999, Hrsg.): Open Sources: Voices of the  
Open Source Revolution, Sebastopol, S. 149 - 170.

BerliOS (Hrsg.) (2001): BerliOS: Der Open-Source-Mediator,  
<http://www.berlios.de>, Abruf am 2001-09-27.

Bernstein, D. J. (2001): Internet host SMTP server survey,  
<http://cr.yp.to/surveys/smtpsoftware5.txt>, 2000-10-05, Abruf am 2001-09-08.

Bezroukov, N. (2001): A Second Look at the Cathedral and the Bazaar,  
[http://www.firstmonday.dk/issues/issue4\\_12/bezroukov/index.html](http://www.firstmonday.dk/issues/issue4_12/bezroukov/index.html), 1999-12-09,  
Abruf am 2001-10-04.

Biethahn, J./Mucksch, H./Ruf, W. (1996): Ganzheitliches Informationsmanage-  
ment, Bd. 1. Grundlagen, 4., durchges. Aufl., München u.a.

Borchers, D. (2001): Lizenzmodelle freier Software,  
<http://www.mikro.org/Events/OS/ref-texte/borchers.html>, Abruf am 2001-10-05.

- Borland Software (Hrsg.) (2001a): Inprise/Borland Introduces InterBase 6.0 Now Free and Open Source on Linux, Windows, and Solaris,  
<http://www.borland.com/about/press/2000/ib6.html>, 2000-07-16, Abruf am 2001-09-28.
- Borland Software (Hrsg.) (2001b): InterBase Public License,  
<http://www.borland.com/devsupport/interbase/opensource/IPL.html>, 2000-07-16, Abruf am 2001-09-28. (Version 1.0).
- Borsook, P. (2001): How Anarchy Works: On location with the masters of the metaverse, the Internet Engineering Task Force,  
<http://www.wired.com/wired/archive/3.10/ietf.html>, Abruf am 2001-09-14.
- Brooks, F. P. J. (1975): The Mythical Man-Month: Essays on Software Engineering, Reading.
- Bundesministerium für Wirtschaft und Technologie (BMWi) (Hrsg.) (2001): Open-Source-Software: Ein Leitfaden für kleine und mittlere Unternehmen,  
<http://www.bmwi.de/Homepage/download/infogesellschaft/Open-Source-Software.pdf>, Abruf am 2001-08-26.
- Collab.Net (Hrsg.) (2001): SourceCast,  
<http://www.collab.net/products/sourcecast/>, Abruf am 2001-09-05.
- Dalheimer, M. (2001): Management eines OpenSource-Softwareprojektes am Beispiel von KDE,  
<http://www.mikro.org/Events/OS/ref-texte/dalheimer.html>, Abruf am 2001-08-08.
- Deutsche Gesellschaft für Qualität (Hrsg.) (1995): Methoden und Verfahren des Qualitätsmanagements für Software, DGQ-ITG-Band 12-52, 2. Aufl., Berlin u.a.
- DiBona, C./Ockman, S./Stone, M. (1999): Introduction.  
In: DiBona, C./Ockman, S./Stone, M. (1999, Hrsg.): Open Sources: Voices of the Open Source Revolution, Sebastopol, S. 1 - 17.
- Duncan, A./Hull, S. (2001): Oracle & Open Source,  
<http://www.oreilly.de/catalog/oracleopen/chapter/ch01.html>, Abruf am 2001-06-11. (Kapitel 1 ist unter der genannten URL abrufbar).
- Eilebrecht, L. (2001): Apache,  
<http://www.mikro.org/Events/OS/ref-texte/eilebrecht.html>, Abruf am 2001-08-09.
- Electronic Frontier Foundation (Hrsg.) (2001): EFF Open Audio License,  
[http://www.eff.org/IP/Open\\_licenses/eff\\_loal.html](http://www.eff.org/IP/Open_licenses/eff_loal.html), 2001-04-21, Abruf am 2001-10-11. (Version 1.0.1).

- Endres, A. (2000): "Open Source" und die Zukunft der Software.  
In: Informatik-Spektrum, Volume 32, Issue 5, 2000, Berlin u.a., S. 316 - 321.
- Ettrich, M. (2001): Wer kodiert? Gedanken zur Freie-Software-Szene,  
<http://www.heise.de/ix/artikel/2000/01/112/>, Abruf am 2001-08-10.
- Flynn, L. J. (2001): New Economy: Open-Source Movement Advances,  
<http://www.nytimes.com/2001/06/04/technology/04NECO.html>, 2001-06-04,  
Abruf am 2001-09-02. (Abruf erst nach Anmeldung möglich).
- Fogel, K. (2000): Open Source-Projekte mit CVS, Bonn.
- Free Software Foundation (Hrsg.) (2001a): Why "Free Software" is better than  
"Open Source",  
<http://www.fsf.org/philosophy/free-software-for-freedom.html>, Abruf am  
2001-08-06.
- Free Software Foundation (Hrsg.) (2001b): The Free Software Definition,  
<http://www.fsf.org/philosophy/free-sw.html>, Abruf am 2001-08-06.
- Free Software Foundation (Hrsg.) (2001c): Free Software Licenses,  
<http://www.fsf.org/licenses/licenses.html>, Abruf am 2001-08-06.
- Free Software Foundation (Hrsg.) (2001d): Categories of Free and Non-Free  
Software,  
<http://www.fsf.org/philosophy/categories.html>, Abruf am 2001-08-07.
- Free Software Foundation (Hrsg.) (2001e): GNU General Public License,  
<http://www.fsf.org/licenses/gpl.html>, Abruf am 2001-08-07. (Version 2).
- Free Software Foundation (Hrsg.) (2001f): GNU Lesser General Public License,  
<http://www.fsf.org/licenses/lgpl.html>, Abruf am 2001-08-07. (Version 2.1).
- Free Software Foundation (Hrsg.) (2001g): Frequently Asked Questions about the  
GNU GPL,  
<http://www.fsf.org/copyleft/gpl-faq.html>, Abruf am 2001-10-10.
- Ghezzi, C./Jazayeri, M./Mandrioli, D. (1991): Fundamentals of Software  
Engeneering, Englewood Cliffs u.a.
- Ghosh, R. A. (2001): Cooking pot markets: an economic model for the trade in free  
goods and services on the Internet,  
[http://www.firstmonday.dk/issues/issue3\\_3/ghosh/index.html](http://www.firstmonday.dk/issues/issue3_3/ghosh/index.html), Abruf am  
2001-08-10.



Glasl, A. (2001): Kalle Dalheimer plaudert über KDE, Linux und Open Source, <http://www.computerwoche.de/info-point/newsdatenbank/details.cfm?SNUMMER=12769>, 1999-11-09, Abruf am 2001-08-02.

Grassmuck, V. (2001a): Geschichte und Mechanismen freier Software, <http://mikro.org/Events/OS/text/gesch-freie-sw.html>, Abruf am 2001-08-01.

Grassmuck, V. (2001b): Offene Quellen und öffentliches Wissen, <http://waste.informatik.hu-berlin.de/Grassmuck/Texts/wos-moskau.html>, Abruf am 2001-08-08.

Grassmuck, V. (2001c): Von Fischteichen, WG-Kühlschränken und freier Software, <http://mikro.org/Events/OS/text/wg-kuehlschraenke.html>, Abruf am 2001-08-08.

Grassmuck, V. (2001d): Die Wissens-Allmende, <http://mikro.org/Events/OS/interface5/wissens-allmende.html>, Abruf am 2001-08-08.

Grochla, E. (1995): Grundlagen der organisatorischen Gestaltung, Stuttgart.

Gulbins, J./Obermayr, K. (1995): UNIX System V.4: Begriffe, Konzepte, Kommandos, Schnittstellen, 4., überarb. Aufl., Berlin u.a.

Hecker, F. (2001): Setting Up Shop: The Business of Open-Source Software, <http://www.hecker.org/writings/setting-up-shop.html>, Abruf am 2001-08-30. (Revision 0.8 vom 2000-06-20).

Hetze, S. (2001): Wirtschaftliche Aspekte von Freier Software und OpenSource, <http://www.telepolis.de/deutsch/special/wos/6465/1.html>, 1999-08-09, Abruf am 2001-09-03.

Heuer, K. (2001): FreeBSD: Die ersten Schritte, <http://gwdu60.gwdg.de/fststeps/>, Abruf am 2001-08-24.

Hill, W./Fehlbaum, R./Ulrich, P. (1994): Organisationslehre 1: Ziele, Instrumente und Bedingungen der Organisation sozialer Systeme, 5., überarb. Aufl., Bern u.a.

Hofmann, J. (2001): Der Erfolg offener Standards und seine Nebenwirkungen am Beispiel der IETF, <http://www.mikro.org/Events/OS/ref-texte/hofmann.html>, Abruf am 2001-10-12.

Hohndel, D. (2001): XFree86, <http://www.mikro.org/Events/OS/ref-texte/hohndel.html>, Abruf am 2001-08-09.

Institute for Telecommunication Sciences (Hrsg.) (2001): American National Standard for Telecommunications – Telecom Glossary 2000,  
<http://its.blrdoc.gov/projects/t1glossary2000/>, Abruf am 2001-08-12.

Kahney, L. (2001): Mexican Schools Embrace Linux,  
<http://www.wired.com/news/technology/0,1282,16107,00.html>, 1998-11-06, Abruf am 2001-08-08.

Keller, G./Nüttgens, M./Scheer, A.-W. (2001): Semantische Prozessmodellierung auf der Grundlage “Ereignisgesteuerter Prozessketten (EPK)”,  
<http://www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/heft089/heft089.pdf>, Abruf am 2001-09-02.

Köppen, A./Nüttgens, M. (2000): Open Source: Strategien für die Beratung.  
In: Scheer, A.-W./Köppen, A. (2000, Hrsg.): Consulting: Wissen für die Strategie-, Prozess- und IT-Beratung, Berlin u.a., S. 231 - 242.

Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik (KBSt) (Hrsg.) (2001): KBSt-Brief Nr. 2/2000: Open Source Software in der Bundesverwaltung,  
<http://www.kbst.bund.de/papers/briefe/02-2000/brief2-2000.html>, Abruf am 2001-08-08.

Lakhani, K./von Hippel, E. (2001): How Open Source software works: “Free” user-to-user assistance,  
<http://web.mit.edu/evhippel/www/opensource.PDF>, Abruf am 2001-10-05.

Lehmann, M. (2001): The GIMP,  
<http://www.mikro.org/Events/OS/ref-texte/lehmann.html>, Abruf am 2001-08-09.

Lerner, J./Tirole, J. (2000): The Simple Economics of Open Source.  
In: National Bureau of Economic Research (2000, Hrsg.): NBER Working Paper Series, Working Paper 7600, Cambridge.

Lerner, J./Tirole, J. (2001): The open source movement: Key research questions.  
In: European Economic Review, Volume 45, Issue 4 - 6, 2001, Amsterdam u.a., S. 819 - 826.

Levy, S. (1994): Hackers: Heroes of the Computer Revolution, 2. Aufl., London.

Linstone, H. A./Turoff, M. (Hrsg.) (1975): The Delphi Method: Techniques and Applications, London u.a.

- Linux Documentation Project (Hrsg.) (2001): Linux Documentation Project (LDP) Manifesto,  
<http://www.linuxdoc.org/manifesto.html>, Abruf am 2001-10-20.
- Maslow, A. H. (1954): Motivation and Personality, New York.
- Maslow, A. H. (1971): The Farther Reaches of Human Nature, New York.
- Mertens, P./Bodendorf, F./König, W./Picot, A./Schumann, M. (1996): Grundzüge der Wirtschaftsinformatik, 4., verb. Aufl., Berlin u.a.
- Metzger, A./Jaeger, T. (2001): Open Source Software and German Copyright Law. In: Max Planck Institute for Foreign and International Patent, Copyright and Competition Law (2001, Hrsg.): International Review of Industrial Property and Copyright Law, Volume 32, No. 1/2001, München, S. 52 - 74.
- Meyer, H. A. (1998): Von Punkt zu Punkt: Skizzen zu einer Theorie der interaktiven Medien.  
In: Nöth, W./Wenz, K. (1998, Hrsg.): Medientheorie und die digitalen Medien, Kassel, S. 177 - 193.
- Müller, M. (1999): Open Source: Standortbestimmung.  
In: O'Reilly & Associates (1999, Hrsg.): Open Source: kurz & gut, Köln, S. 7 - 19.
- Mundie, C. (2001): The Commercial Software Model,  
<http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.asp>,  
2001-05-03, Abruf am 2001-09-02.
- Netcraft (Hrsg.) (2001): Netcraft Web Server Survey,  
<http://www.netcraft.com/survey/>, Abruf am 2001-09-03.
- Nüttgens, M. (2001): Open Source: Anwendungsentwicklung im Internetzeitalter,  
[http://www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/Gi-fa\\_51/Gi-fa\\_51.pdf](http://www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/Gi-fa_51/Gi-fa_51.pdf), Abruf am 2001-08-03.
- Nüttgens, M./Tesei, E. (2001a): Open Source: Konzept, Communities und Institutionen,  
<http://www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/heft156/heft156.pdf>, Abruf am 2001-08-03.
- Nüttgens, M./Tesei, E. (2001b): Open Source: Produktion, Organisation und Lizenzen,  
<http://www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/heft157/heft157.pdf>, Abruf am 2001-08-03.

Nüttgens, M./Tesei, E. (2001c): Open Source: Marktmodelle und Netzwerke,  
<http://www.iwi.uni-sb.de/nuettgens/Veroef/Artikel/heft158/heft158.pdf>, Abruf am  
2001-08-03.

OpenContent (Hrsg.) (2001a): OpenContent,  
<http://www.opencontent.org>, Abruf am 2001-10-09.

OpenContent (Hrsg.) (2001b): OpenContent License,  
<http://opencontent.org/opl.shtml>, Abruf am 2001-10-09.

OpenContent (Hrsg.) (2001c): Open Publication License,  
<http://opencontent.org/openpub/>, Abruf am 2001-10-09.

OpenMusic Initiative (Hrsg.) (2001): The OpenMusic License,  
<http://openmusic.linuxtag.org/showitem.php?item=209>, Abruf am 2001-10-11.

Open Source Initiative (Hrsg.) (2001): The Open Source Definition,  
<http://www.opensource.org/docs/definition.html>, Abruf am 2001-08-05.  
(Version 1.8).

Open Theory (Hrsg.) (2001): Open Theory,  
<http://www.opentheory.org>, Abruf am 2001-10-05.

O'Reilly & Associates (Hrsg.) (1999): Open Source-Projekte.  
In: O'Reilly & Associates (1999, Hrsg.): Open Source: kurz & gut, Köln, S. 21 - 31.

O'Reilly, T. (2001): Schlüsse aus der Open-Source-Software-Entwicklung,  
<http://www.heise.de/tp/deutsch/special/wos/6433/1.html>, 1999-07-13, Abruf am  
2001-08-10.

Perens, B. (2001): Debian GNU/Linux: A Social Contract,  
[http://www.debian.org/social\\_contract.html](http://www.debian.org/social_contract.html), Abruf am 2001-08-07.

Perl Mongers (Hrsg.) (2001): Perl Fast Facts,  
[http://www.perl.org/press/fast\\_facts.html](http://www.perl.org/press/fast_facts.html), Abruf am 2001-08-10.

Raymond, E. S. (2001a): How to Become a Hacker.  
In: Raymond, E. S. (2001, Hrsg.): The Cathedral and the Bazaar: Musings on Linux  
and Open Source by an Accidental Revolutionary, Revised Edition, Sebastopol,  
S. 195 - 213.

Raymond, E. S. (2001b): A Brief History of Hackerdom.  
In: Raymond, E. S. (2001, Hrsg.): The Cathedral and the Bazaar: Musings on Linux  
and Open Source by an Accidental Revolutionary, Revised Edition, Sebastopol,  
S. 1 - 17.

Raymond, E. S. (2001c): Homesteading the Noosphere.

In: Raymond, E. S. (2001, Hrsg.): The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Revised Edition, Sebastopol, S. 65 - 111.

Raymond, E. S. (2001d): The Cathedral and the Bazaar.

In: Raymond, E. S. (2001, Hrsg.): The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Revised Edition, Sebastopol, S. 19 - 63.

Raymond, E. S. (2001e): Eric Steven Raymond's Home Page,

<http://www.tuxedo.org/~esr/>, Abruf am 2001-08-19.

Raymond, E. S. (2001f): Notes, Bibliography, and Acknowledgments.

In: Raymond, E. S. (2001, Hrsg.): The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Revised Edition, Sebastopol, S. 219 - 235.

Raymond, E. S. (2001g): The Magic Cauldron.

In: Raymond, E. S. (2001, Hrsg.): The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary, Revised Edition, Sebastopol, S. 113 - 166.

Ronneburg, F. (2001): Debian GNU/Linux,

<http://www.mikro.org/Events/OS/ref-texte/ronneburg.html>, Abruf am 2001-08-09.

Salus, P. H. (1994): A Quarter Century of UNIX, Reading u.a.

SAP (Hrsg.) (2001): SAP Drives Open-Source Database Development,

<http://www.sap.com/company/press/press.asp?pressID=99>, 2000-10-05, Abruf am 2001-09-25.

Scheer, A.-W. (1992): Architektur integrierter Informationssysteme: Grundlagen der Unternehmensmodellierung, 2. Aufl., Berlin u.a.

Schierenbeck, H. (1973): Beteiligungsentscheidungen: Betriebswirtschaftliche Grundlagen des Erwerbs und der Veräußerung von Unternehmensbeteiligungen, Berlin.

Schulze, G. (2001): Meine Rechte als Urheber: Urheber- und Verlagsrecht, 4., aktual. Aufl., München.

Schwander, P. (1996): Qualitätsmanagement in Software-Entwicklungsunternehmen, Als Ms. gedr., Aachen.

- Shulgin, A. (2001): Open Sources in Net Art,  
<http://www.mikro.org/Events/OS/ref-texte/shulgin.html>, Abruf am 2001-10-04.
- Siepmann, J. (2001): Lizenz- und haftungsrechtliche Fragen bei der kommerziellen Nutzung Freier Software,  
<http://www.jura.uni-sb.de/jurpc/aufsatz/19990163.htm>, Abruf am 2001-10-12.
- Software in the Public Interest (Hrsg.) (2001): Software in the Public Interest,  
<http://www.spi-inc.org>, Abruf am 2001-09-02.
- Stallman, R. (1999): The GNU Operating System and the Free Software Movement.  
In: DiBona, C./Ockman, S./Stone, M. (1999, Hrsg.): Open Sources: Voices of the Open Source Revolution, Sebastopol, S. 53 - 70.
- Stallman, R. (2001a): Richard Stallman's Personal Home Page,  
<http://www.stallman.org>, Abruf am 2001-08-10.
- Stallman, R. (2001b): Free Software and Beyond,  
<http://www.mikro.org/Events/OS/ref-texte/stallman.html>, Abruf am 2001-10-01.
- Stallman, R. (2001c): Initial Announcement,  
<http://www.fsf.org/gnu/initial-announcement.html>, 1983-09-27, Abruf am 2001-08-12.
- Stutz, M. (2001): Design Science License,  
<http://dsl.org/copyleft/dsl.txt>, Abruf am 2001-10-09.
- Tanenbaum, A. S. (1987): Operating Systems: Design and Implementation,  
Englewood Cliffs u.a.
- The Apache Software Foundation (Hrsg.) (2001a): The Apache Software License,  
<http://www.apache.org/LICENSE.txt>, Abruf am 2001-08-10. (Version 1.1).
- The Apache Software Foundation (Hrsg.) (2001b): About the Apache HTTP Server Project,  
[http://www.apache.org/ABOUT\\_APACHE.html](http://www.apache.org/ABOUT_APACHE.html), Abruf am 2001-08-10.
- The Apache Software Foundation (Hrsg.) (2001c): The Apache Software Foundation,  
<http://www.apache.org>, Abruf am 2001-08-10.
- The K Desktop Environment (Hrsg.) (2001a): The KDE Development Model,  
<http://www.kde.org/whatiskde/devmodel.html>, Abruf am 2001-09-03.
- The K Desktop Environment (Hrsg.) (2001b): KDE Bug Tracking System,  
<http://bugs.kde.org>, Abruf am 2001-09-20.

The Library of Congress (Hrsg.) (2001): Copyright Law of the United States of America,

<http://www.loc.gov/copyright/title17/circ92.html>, Abruf am 2001-08-26.

The Mozilla Organization (Hrsg.) (2001a): Mozilla Public License,

<http://www.mozilla.org/MPL/MPL-1.1.html>, Abruf am 2001-08-10. (Version 1.1).

The Mozilla Organization (Hrsg.) (2001b): Netscape Public License,

<http://www.mozilla.org/MPL/NPL-1.1.html>, Abruf am 2001-08-10. (Version 1.1).

Trolltech (Hrsg.) (2001): The Q Public License,

<http://www.trolltech.com/products/download/freelicense/plaintext.html>, Abruf am 2001-08-10. (Version 1.0).

VA Linux Systems (Hrsg.) (2001a): SourceForge.net,

<http://sourceforge.net>, Abruf am 2001-09-02.

VA Linux Systems (Hrsg.) (2001b): SourceForge Portal Edition,

[http://www.valinux.com/sf/sf\\_portal.php](http://www.valinux.com/sf/sf_portal.php), Abruf am 2001-09-08.

Valloppillil, V. (2001): Open Source Software: A (New?) Development Methodology,

<http://www.opensource.org/halloween/halloween1.html>, Abruf am 2001-08-21.

Weber, V. (2001): "Wir teilen uns das Wasser und die Müsliriegel": Ein Interview mit Jonathan L. Prial, IBM Director Integrated Solutions and Linux Marketing, über UNIX, Windows 2000 und die Position von Linux,

<http://www.heise.de/ct/99/17/046/>, Abruf am 2001-08-10.

Wheeler, D. A. (2001): More Than a Gigabuck: Estimating GNU/Linux's Size,

<http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.html>, 2001-06-30, Abruf am 2001-10-05. (Version 1.04).

Yeh, H.-T. (1993): Software process quality, New York u.a.

Young, M. (1961): Es lebe die Ungleichheit: Auf dem Wege zur Meritokratie, Düsseldorf.

Zawinski, J. (2001): resignation and postmortem,

<http://www.jwz.org/gruntle/nomo.html>, 1999-03-31, Abruf am 2001-09-17.

# Anhang A:

## GNU Free Documentation License

Version 1.1, March 2000

Copyright © 2000 Free Software Foundation, Inc.

59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### 0. Preamble

The purpose of this License is to make a manual, textbook, or other written document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

### 1. Applicability and Definitions

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The



“Document”, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as “you”.

A “Modified Version” of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format,  $\text{\LaTeX}$  input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title,

preceding the beginning of the body of the text.

## 2. Verbatim Copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. Copying in Quantity

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute

an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified

Version as given on the Title Page. If there is no section entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled “Acknowledgements” or “Dedications”, preserve the section’s title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled “Endorsements”. Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as “Endorsements” or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties – for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may

not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## **5. Combining Documents**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled “History” in the various original documents, forming one section entitled “History”; likewise combine any sections entitled “Acknowledgements”, and any sections entitled “Dedications”. You must delete all sections entitled “Endorsements.”

## **6. Collections of Documents**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. Aggregation With Independent Works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an “aggregate”, and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document’s Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

## 8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## 9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. Future Revisions of This License

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit

to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License “or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## **ADDENDUM: How to use this License for your documents**

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

Copyright © YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST. A copy of the license is included in the section entitled “GNU Free Documentation License”.

If you have no Invariant Sections, write “with no Invariant Sections” instead of saying which ones are invariant. If you have no Front-Cover Texts, write “no Front-Cover Texts” instead of “Front-Cover Texts being LIST”; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

# Erklärung

Ich versichere, dass ich die Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen oder anderen Quellen entnommen sind, sind als solche kenntlich gemacht.

Frank Bauer