Entwickler Freier Software als soziale Formation
Ethnographie und Fallstudie

Frauke Lehmann

Oktober 2004

Der vorliegende Text ist eine nicht substantiell geänderte Version meiner Magisterarbeit bei PD Dr. Thomas Zitelmann und PD Dr. Jörg Strübing am Institut für Ethnologie an der Freien Universität Berlin aus dem Oktober 2004. Der Text kann unter http://userpage.fu-berlin.de/~egal/download/Frauke_Lehmann-Entwickler_Freier_Software_als_soziale_Formation.pdf heruntergeladen werden.

Danksagung

Ich möchte mich hier bei allen Personen bedanken, die es mir ermöglicht haben, diesen Text zu schreiben, sei es durch betreuende Unterstützung, sei es durch Informationen und Interviews, sie es durch weiterbringende Fragen oder Anmerkungen.

Mein Dank gilt:

Holger Blasum, Andreas Brandt, Manuel Borchers, Oswald Buddenhagen, Benedikt Christopeit, Georg Elwert, Benjamin Franksen, Tilo Fuchs, Georg Greve, Jon „Maddog“ Hall, Sven Herzberg, Andreas Frank Hoffmann, Jörg Hoh, Kurt Jansson, Nicolas Kratz, Utz Lehmann, Dani Oderbolz, Marc 'BlackJack' Rintsch, Michael Scharkow, Stefan Schumacher, Christian Siefkes, Jörg Strübing, Thomas Templin, Andreas Thienemann, Daniel Tippmann, David Turner, Holger Weiss, Harald Welte, Arne Wichmann, Maximilian Wilhelm, Enrico Zini, Thomas Zitelmann und natürlich alle, die sich an der Umfrage beteiligt haben.

Inhaltsverzeichnis

Abkürzungen

AI Lab Artificial Intellegence Lab

BSD Berkeley Software Distribution

CCC Chaos Computer Club

DPL Debian Projekt Leader

FLI4L Floppy ISDN for Linux

FSF Free Software Foundation

Gnome GNU Network Object Model Environment

GNU GNU is not Unix

GPL GNU General Public License

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IRC Internet Relay Chat

KDE K Desktop Enviroment

LGPL Gnu Lesser Gerneral Public License

MIT Massachusetts Institute of Technology

SCO Santa Cruz Operation

SLS Softlanding Linux System

SPI Software in the Public Interest

TCP/IP Transmission Control Protocol/Internet Protocol

UNIX Uniplexed Information and Computing Service

URL Universal Resource Identifier (ursprünglich URI)

USL Unix System Laboratories

UUCP Unix toUnix Copy Protocol

WWW World Wide Web

Kapitel 1

Einleitung

Karl-Heinz Kohl hat die Ethnologie als Wissenschaft des kulturell Fremden bezeichnet. Er versucht mit dem Begriff des *kulturell Fremden* die unterschiedlichsten menschlichen Zusammenschlüsse zusammenzufassen, die für sich genommen durchaus heterogen sein können und nur in Bezug auf unsere eigene Kultur als eine homogene Einheit erscheinen. Da nach Kohl nicht alle fremden Kulturen zum traditionellen Gegenstandsbereich der Ethnologie gehören, führt er den Grad der Unterschiedlichkeit ein: „Je mehr sich eine bestimmte Lebensgemeinschaft von Menschen unserer eigenen Kultur unterschied, desto eher schien sie dazu prädestiniert, zum Gegenstand ethnologischer Forschung zu werden“ (Kohl 1993: 26). Bis in die erste Hälfte des 20. Jahrhunderts wurden immer neue Völker „entdeckt”. Ein wichtiger Hintergrund des Erkenntnisinteresses war der Gedanke in diesen „primitiven“ Völker die eigene Vergangenheit und so das Werden und das Entwicklungsschema der gesellschaftlichen Menschheit zu erkennen (Kohl 1993: 19ff.). Seitdem ist einige Zeit vergangen. Heute beschäftigt sich die Ethnologie mit den unterschiedlichsten Gruppierungen: so genannten Small-Scale-Societies, der Verbindung von Bauern in Indien mit globalen Organisationen wie dem Internationalen Währungsfond, oder auch mit Gruppen vor unserer Haustür, die durch ihre Nationalität, ihre religiöse Dazugehörigkeit usw. eine Minderheit innerhalb der Gesellschaft bilden. Also beschäftigt sich die Ethnologie mit den verschiedensten Arten von sozialen Formationen. Insbesondere sind es solche, die nicht zur Mehrheitsgesellschaft gehören. Hinzu kommen globale Netzwerke und Bewegungen. In diesem Bereich ist auch die in dieser Untersuchung betrachtete soziale Formation anzusiedeln: Menschen, die Freie Software entwickeln.[1]

[1] Freie Software zeichnet sich durch einen freigegebenen Quellcode aus. Dieser ist die Voraussetzung, dass andere diese Software nach ihrem Belieben ändern können. Eine genauere Klärung was dies alles beinhaltet, wird im Laufe dieser Arbeit gegeben werden.

Entwickler Freier Software sind hier nicht nur die Programmierer (Codeschreiber), sondern auch Dokumentierer, Übersetzer, Personen, die sich um die Infrastruktur kümmern (Server, Webseiten, Mailinglisten usw. pflegen) und Öffentlichkeitsarbeit (Userbetreuung) betreiben. Reine Nutzer werden nicht dazu gezählt. Sie gehören zwar zum Bereich Freie Software, tragen aber eigentlich nicht

In der folgenden Arbeit soll geklärt werden, um was für eine Art des sozialen Zusammenhangs es sich bei den Entwickler von Freier Software handelt. Die Entwickler sind nicht nur über die ganze Welt verteilt, ihre Beteiligung an der Produktion Freier Software ist auch zunächst einmal ortsunabhängig – so lange Computer und Internetzugang vorhanden sind. Die Produzenten beteiligen sich alle aus freien Stücken. Anders als beispielsweise bei globalen Unternehmen sind vorab keine Strukturen und Regeln der Zusammenarbeit vorgegeben. Daher müssen sich die Entwickler Freier Software selbst Integrationsmechanismen schaffen. Dies ist auf der einen Seite – was den Umfang betrifft – ein relativ neues Phänomen. Zwar haben schon früher Menschen zusammengearbeitet, die über die ganze Welt verstreut waren, doch dies war aufgrund der fehlenden Kommunikationstechnologie nur in geringem Ausmaß möglich. Auf der anderen Seite hat dieses Phänomen sehr wohl bekannte Wurzeln: die Beteiligten kommen aus einer bestimmten „Hackerkultur“[2], die sich bis in die sechziger Jahre zurückverfolgen lässt – wenn nicht noch weiter. So gab es auch schon Vorstellungen, wie diese neue, globale Formation aussehen könnte und Traditionen, auf die sie aufbauen konnte.

Um diese neue soziale Zusammenschlüsse fassen zu können, sollen zunächst verschiedene theoretische Modelle sozialer Formation betrachtet werden, die auf den ersten Blick angemessen erscheinen, um die untersuchte soziale Formation genauer bestimmen zu können: Gemeinschaft, Gruppe, Organisation, Netzwerk, soziale Bewegung und soziale Welt. Hierdurch sollen Vergleichsmaßstäbe bzw. Analysehilfen für die Verortung der Entwickler Freier Software als soziale Formation geliefert werden.

Die Untersuchung des Feldes beginnt mit der Beschreibung wie dieses Phänomen entstanden ist. Aus welchen anderen Feldern hat es sich entwickelt? Welche Aspekte wurden woher übernommen? Was waren die jeweiligen äußeren Umstände, die die Entwicklung beeinflusst haben? Und welcher Wandel vollzog sich seit der Entstehung? Neben der Untersuchung der geschichtlichen Entwicklung soll dieser Teil die Einführung in das Feld leisten.

Danach sollen die Grenzen herausgearbeitet werden, beginnend mit der Frage, was *Freie* Software eigentlich heißt. Die Antwort findet sich auf der eigentumsrechtlichen Ebene. Anschließend wird eine weitere Grenzziehung betrachtet, nämlich die Abgrenzung der Entwickler Freier Software von anderen durch ihr Selbstbild. Dieses umfasst Elemente wie Normen, Lebensstile und Klischees. Diese Form der subjektiven Abgrenzung findet sich auch innerhalb des Feldes zwischen den Projekten.

---

zur Entwicklung bei, außer wenn sie Fehlermeldungen einschicken. Mittlerweile sind auch einige Unternehmen in diesem Feld tätig, auch ihre Rolle soll aus diese Arbeit nicht behandelt werden, da dies den Rahmen sprengen würde.

Der häufig benutzte Begriff „Open Source“ ist mit dem Begriff *Freie Software* identisch. Eine Erläuterung, warum es beide Begriffe gibt, wird ebenfalls in der Arbeit folgen.

[2]Mit dem Begriff Hacker sind hier nicht, wie es mittlerweile umgangssprachlich weit verbreitet ist, Menschen gemeint, die in fremde Rechner einbrechen, dort unberechtigt Daten einsehen, Schäden anrichten, etc., sondern Menschen, die aus einer Leidenschaft heraus programmieren.

Das nächste Kapitel wird sich mit der Struktur des Feldes befassen. Einführend soll auf die allgemeine Rahmenbedingungen und die Kommunikationsweisen der Projekte eingegangen werden. Anhand einiger Fallbeispiele soll danach die thematische und organisatorische Vielfalt aufgezeigt werden. Im Anschluss findet eine Analyse der Projektstrukturen statt. Es soll ermittelt werden, wie sich ihre Strukturen herausbilden. Aspekte wie Heterogenität, Fragmentierung, Organisationsformen und die Entscheidungsstrukturen sollen genauer untersucht werden.

Im Anschluss soll es um die Prozesse der sozialen Kohäsion innerhalb des Feldes gehen. Zunächst wird die personelle Ebene beleuchtet werden. Hierzu wird zunächst eine Beschreibung der Menschen erfolgen, die sich an der Entwicklung Freier Software beteiligen, dann wird auf die Motivation zur Beteiligung eingegangen werden. Auf der Ebene der Projekte soll dann ermittelt werden, wie diese langfristige Stabilität erreichen. Dazu sollen Formen der Konfliktlösung, der Enkulturation und sonstige stabilisierende Mechanismen betrachtet werden.

Mit Bezug zu den am Beginn der Arbeit vorgestellten Ansätzen soll dann versucht werden, die gemachten Beobachtungen aufzuarbeiten und so zu einer theoretischen Einordnung des Feldes zu kommen.

Den Abschluss der Arbeit bildet die Frage, welche Bedeutung dieses Feld für die Ethnologie hat. Es ist ein für das Fach ungewöhnliches Feld, das durch seine Determinanten unter Umständen einen interessanten und wichtigen Diskussionsbeitrag für die theoretische Konzeption des gesellschaftlichen Zusammenlebens und der Herausbildung neuer Gemeinschaftsformen sein könnte.

Methoden

Die Daten dieser Arbeit haben eine sehr unterschiedliche Herkunft. Zum einem wurde nach einer kritischen Prüfung auf die bereits existierende Literatur aufgebaut. Zum anderen wurden eigene Daten erhoben.

Die sozialen Prozesse zwischen den Entwicklern finden größtenteils im Internet statt, also einem virtuellen Raum. Dadurch sind die klassischen Methoden der Ethnographie, vor allem die teilnehmenden Beobachtung, nur sehr begrenzt einsetzbar. Die Größe, Heterogenität und Ortlosigkeit des Feldes erschweren den Zugang und das Identifizieren relevanter Personen und Prozesse. Verschärfend wirkt sich aus, dass nicht-technische Themen und Fragen im hauptsächlichen Kommunikationsraum – den Mailinglisten – verpönt sind. Daher mussten die gewählten Methoden diesen Umständen angepasst werden: Beobachtungen auf realweltlichen Treffen, Interviews und sonstige Unterhaltungen mit verschiedenen Mitgliedern des Felds, eine online-Umfrage unter Entwicklern, Analysieren von Webseiten diverser Projekte, internen Texten und einigen Projekt-Mailinglisten.

Eine wichtige Voraussetzung, um die Geschehnisse im Feld verstehen zu können ist ein Mindestmaß am Verständnis der technischen Materie. Das bedeutet nicht, dass man zwingenderweise über Programmierkenntnisse verfügen muss, sondern eher, dass man in der Lage ist, die groben Zusammenhänge nachvollziehen zu können. Bei der technischen Einarbeitung waren mir hauptsächlich Nutzer Freier Software aus meinem privaten Umfeld behilflich, desweiteren benutze ich selbst seit fast zwei Jahren die betreffende Software.

Die realweltliche Beobachtungen fanden hauptsächlich auf dem Linuxtag in Karlsruhe in den Jahren 2002, 2003 und 2004, dem Linuxtag in Magdeburg 2003 und bei Wizards of OS 3 in Berlin 2004 statt.[3] Dort führte ich zum einen Interviews mit Mitarbeitern diverser Projekte, sonstigen Entwicklern und Mitarbeiter verschiedener feldrelevanter politischer Organisationen, wie der Free Software Foundation Europe und dem Förderverein für eine Freie Informationelle Infrastruktur e. V. (FFII). Zum anderen war ich für diese Zeit fast dauernd mit den Entwicklern zusammen und habe an verschiedenen nicht-technischen Veranstaltungen und Parties teilgenommen, was für das Verständnis des Feldes sehr hilfreich war. Die Besuche des Linuxtages waren für die nachfolgenden Untersuchungsabschnitte – einschließlich der späteren Linuxtage – unerlässlich. Dort ergab sich immer wieder die Möglichkeit zu Gesprächen, die über Emailkontakt kaum hätten geführt werden können, zu Beobachtungen des Verhaltens der Entwickler untereinander und dazu, meine bisherigen Erkenntnisse mit Angehörigen des Feldes durchzusprechen und sie gegebenenfalls zu relativieren.

Das Ziel des Fragebogens war keine aufwändige statistische Auswertung, um zu allgemeingültigen Aussagen zu kommen. Es ging vielmehr darum, mögliche blinde Flecken zu ermitteln und so Hinweise für die sonstige Arbeit zu gewinnen. Zudem sollten durch geschlossene Fragen persönliche Daten der Entwickler abgefragt werden (Beruf, Ausbildung, Einkommen etc.). Derartige Daten lassen sich im Feld kaum anders erheben als durch Online-Fragebögen, da eine persönliche Anschauung ja kaum möglich ist und Analysen der Kommunikation etc. darüber auch keine Auskunft geben. Ähnlich verhält es sich mit Fragen allgemeinerer Natur, wie die nach der Motivation der Entwickler, der Zeit, die sie für Freie Software aufwenden etc.. Diese werden in Mailinglisten oder auf Webseiten – also den zugänglichen Quellen des Feldes – nicht thematisiert. Interviews wiederum liefern nur Daten zu sehr wenigen Fällen, zudem mit einer hohen regionalen Konzentration. Der Fragebogen wurde in Zusammenarbeit mit Entwicklern und feldnahen Personen entwickelt, um die Relevanz und Richtigkeit der Fragen sicherzustellen. Er enthielt 54 Fragen, die von 223 Entwicklern in der Zeit vom 21.6. bis 16.9.2004 beantwortet wurden. Der Hinweis auf die Befragung wurde erfolgreich an 48 Mailinglisten von Projekten mit unterschiedlichsten Charakteristika versendet. Problematisch bei der Verwendung

[3]Der Linuxtag in Karlsruhe ist nach Angaben des Veranstalters die wichtigste Veranstaltung zum Thema Linux und Freie Software in Europa. Vgl. http://www.linuxtag.de/2005/index.php am 10.10.2004.

eines Online-Fragebogens sind natürlich die Selbstrekrutierung und die unbekannte Grundgesamtheit. Daher sind die Relevanz und Repräsentativität der Stichprobe nicht feststellbar. Allerdings lässt sich im Vergleich mit anderen Umfragen sagen, dass die Zahl der Antworten gut im Mittelfeld liegt. Außer (bisher) zwei Umfragen mit 2.800 bzw. 5.400 Teilnehmern, haben die meisten Samples nur eine Größe von 70-140 (vgl. Luthiger 2004: 103f.; weitere Umfragen laufen). Die Überschneidung vieler Fragenbereiche machte es zudem möglich, die Trends meiner eigenen Ergebnisse mit denen anderer Studien zu vergleichen (insb. Ghosh et al. 2002).

Die Beobachtung von Mailinglisten diente mir vor allem dazu, meine gewonnenen Erkenntnisse anhand beobachteter Fälle zu überprüfen und meine Annahmen über das Feld entsprechend zu verändern. Es wurde keine systematische Analyse durchgeführt, sondern es wurden nur einzelne Beträge und Diskussionsstränge beobachtet. Es wurden folgende Mailinglisten verfolgt: debian-devel@lists.debian.org (ab den 20.4.2004), debian-legal@lists.debian.org (22.10.2003 - 20.9.2004), debian-mentors@lists.debian.org (ab den 20.4.2004), debian-policy@lists.debian.org (22.10.2003 - 20.9.2004), debian-project@lists.debian.org (ab den 22.10.2003), gimp-developer@lists.xcf.berkeley.e (ab den 21.5.2004), desktop-devel-list@gnome.org (ab den 18.4.2004), gnome-devel-list@ gnome.org (ab den 11.8.2004), gnome-de@gnome.org (ab den 18.4.2004), linux-kernel@ vger.kernel.org (ab den 21.5.2004), current-users@NetBSD.org (8.11.2003 - 11.5.2004), tech-kern@NetBSD.org (an den 8.11.2003), misc@openbsd.org (9.11.2003 - 24.4.2004).

Natürlich wurde auch auf vorhandene Literatur zum Thema zurückgegriffen. Diese besteht zu einem großen Anteil aus journalistischen Reportagen über Freie Software und die Geschichte der Computer, sowie Biographien einzelner Entwickler. Ein weiterer Typ sind Texte von Entwicklern, die ihre eigene Arbeit aber auch Prozesse im Feld als Ganzes aufarbeiten und reflektieren, teilweise mit wissenschaftlichem Anspruch. Schließlich gibt es eine wachsende Zahl wissenschaftlicher Arbeiten zum Thema. Einige davon behandeln allgemeine ökonomische Fragen und den gesellschaftlichen Vorbildcharakter des Feldes. Andere konzentrieren sich eher auf Freie Software als Managementprinzip und Geschäftsidee. Zudem entsteht derzeit eine Reihe von Studien über einzelne Softwareprojekte wie auch über das Feld Freie Software insgesamt, es sind aber erst sehr wenige dieser Ergebnisse publiziert.

Kapitel 2

Modelle sozialer Formation

Im folgenden Kapitel werden sieben Konzepte vorgestellt, die sich aus unterschiedlichen Perspektiven mit sozialen Zusammenschlüsse beschäftigen: Gemeinschaft, soziale Gruppe, Organisation, Netzwerk, soziale Bewegung, Subkultur und soziale Welt. Es wird nicht davon ausgegangen, dass ein einzelnes Konzept auf das vorliegende Phänomen vollständig passt. Vielmehr können alle diese Modelle einen Beitrag zu seiner Erklärung liefern, dabei helfen, Unterschiede zu anderen Feldern herauszuarbeiten und schließlich auf weiter zu verfolgende Fragestellungen aufmerksam machen.

In der Diskussion um Freie Software wird diese meist als Gemeinschaft oder auch Community bezeichnet. Daher soll hier zunächst geklärt werden, mit welchen theoretischen Annahmen dieser Begriff aufgeladen ist und welche Art von sozialer Formation hier gemeint ist. Die Konzepte soziale Gruppe, Organisation und Netzwerk sollen bei der inneren Analyse des Feldes helfen. Soziale Gruppe und Organisation können wichtige Hilfen zur Analyse der inneren Strukturierung bieten. Das Konzept des sozialen Netzwerkes soll hier ähnlich verwendet werden. Ein vollständiges Aufzeichnen des Netzwerkes der Entwickler Freier Software ist hier nicht von Bedeutung und auch nicht leistbar, viel mehr sollen einzelne Elemente der Netzwerkanalyse für die qualitiative Beschreibung der Bindungen herausgearbeitet werden. Oft wird das Phänomen Freie Software auch als soziale Bewegung bezeichnet. In diesem Falle wird auf den politischen Gehalt des Phänomens angespielt. Um zu überprüfen, ob dies gerechtfertigt ist, soll ebenso wie beim Begriff der Gemeinschaft zunächst geklärt werden, was dieses Konzept eigentlich beinhaltet. Die Beschäftigung mit den Überlegungen zur Subkultur soll eine bisher vernachlässigte Perspektive in die Diskussion bringen: gibt es ein ähnliches Weltbild der Beteiligten, und wenn ja, worauf begründen es sich? Die Theorie der sozialen Welt geht in ihrer Betrachtung dagegen von der Tätigkeit der Akteure aus, Strukturen und Bindungen sind eher sekundär.

Die Konzepte können hier nur sehr verkürzt behandelt werden. Es geht dabei auch nicht darum, den einzelnen Theorien in ihrer ganzen Tiefe gerecht zu werden, son-

dern um eine Erstellung von Instrumenten für die Untersuchung und die anschließend folgende Analyse der Frage nach der Gestalt der sozialen Formation der Entwickler Freier Software.

2.1 Gemeinschaft

Bei der Beschäftigung mit dem Thema Gemeinschaft stößt man immer wieder auf den Namen Ferdinand Tönnies (vgl. Hillmann 1994 :269). Er erarbeitet seinen Gemeinschaftsbegriff in Gegenüberstellung zum Begriff der Gesellschaft (Tönnies 1991). Gemeinschaft sieht er als ein organisches Verhältnis an. Eine organische Beziehung zeichnet sich durch eine Ganzheit aus, die die einzelnen Elemente nur durch die Verbindung mit den restlichen Bestandteilen erlangen. Mit der Gemeinschaft, in der man sich von Geburt an befindet, verbindet Tönnies das vertraute, heimische, ausschließliche Zusammenleben.[1] Die Gesellschaft dagegen zeichnet sich durch ein bloßes Nebeneinander von unabhängigen Menschen aus. Die Verbindung in der Gesellschaft zwischen den Menschen ist durch Verträge geregelt. Der Einzelne strebt nach Gewinn, den der durch Tausch erhalten kann.[2]

Neben dem Vergleich von Gemeinschaft und Gesellschaft beschäftigt sich Tönnies mit dem Konzept des Willens. Dieses ist wichtig für das Verständnis seines Gemeinschaftsbegriffes. Er teilt den Willen in den Wesenswillen und den Kürwillen. Den ersten ordnet er der Gemeinschaft und den zweiten der Gesellschaft zu.[3] Tönnies sieht die Verankerung des Wesenswillen im Körperlichen, im Empfangen von Reizen (ebd.: 77ff. [2. Buch § 5, 6]). Die Weiterentwicklung des Willens endet im Abstrakten, zum Beispiel die Fähigkeit Sprache zu verstehen und nicht nur den Sprechakt zu reproduzieren. Die verschiedenen Stufen des Wesenswillen beschreibt Tönnies mit Gefallen, Gewohnheit und Gedächtnis. Der Gefallen findet auf einer rein instinktiven Ebene statt, Gewohnheit beruht auf Erfahrung und Gedächtnis ist nötig, um Eindrücke zu reproduzieren und sie gezielt einsetzen zu können, beispielsweise beim Sprechen (ebd.: 82ff. [2. Buch § 8]).

Der Zusammenhalt der Gemeinschaft geschieht durch Verständnis (ebd.: 17 [1. Buch § 9]). Es ist der eigene Wille der Gemeinschaft, die gemeinsam geteilte, verbindende Gesinnung. Dieses Verständnis kann auch als der Sinn einer Gemeinschaft begriffen

[1] Es ist anzumerken, dass der englische Begriff *community* eine etwas anders gelagerte Bedeutung hat. Der Aspekt der organischen Verbindung scheint hier weniger stark ausgeprägt zu sein. Zudem scheint das Wort in viel größerem Maße und mit vielschichtigen Bedeutungen in der Umgangssprache verwendet zu werden.

[2] Tönnies Gesellschaftsbild erinnert stark an Karl Marx (vgl. Marx 1974), auf den er sich teilweise auch selbst bezieht, wenn er auch nicht alle Auffassungen von Marx teilt (Tönnies 1991: 69 f. [1. Buch § 40]). Allerdings soll das Konzept der Gesellschaft hier nicht weiter verfolgt werden, da es nicht im zentralen Untersuchungsfeld dieser Arbeit liegt.

[3] Daher wird der Kürwillen hier auch nicht weiter beachtet werden.

werden. Tönnies zufolge ist sein wahres Organ die Sprache. Wie bei anderen Ausdrucksformen ist ihre Äußerung die unwillkürliche Folge tiefer Gefühle, und dient nicht der Absicht sich *verständlich* zu machen. Sprache ist aus Trautheit, Innigkeit und Liebe entsprungen (ebd.: 17f. [1. Buch § 9]).

Die Gemeinschaft baut Tönnies auf drei Urverhältnisse auf (ebd.: 7f. [1. Buch § 1]):

1. Das Mutter-Kind Verhältnis. Dieses ist am tiefsten durch reinen Instinkt oder auch Gefallen geprägt. Später, bei zunehmenden Alter des Kindes, wird der Instinkt durch Gewöhnung und Gedächtnis ersetzt.

2. Das Verhältnis zwischen Mann und Frau. Dieses Verhältnis entsteht hauptsächlich durch Gewöhnung. Zur Befriedigung des Sexualinstinkts, ist, nach Tönnies, kein dauerhaftes Zusammenleben nötig, erst die Gewöhnung gestaltet dieses Verhältnis dauerhaft. Gemeinsame Kinder und sonstiger (ebd.: 8 [1. Buch §1]) gemeinsamer Besitz wirken als stabilisierende Faktoren.

3. Das Verhältnis unter Geschwistern (sich als Kinder der gleichen Mutter Kennende). Bei diesem Verhältnis gibt es kein ursprüngliches und instinktives Gefallen. Tönnies sieht es als die am meisten menschliche und doch auf Blutsverwandtschaft beruhende Beziehung an, da hier der Instinkt das schwächste und das Gedächtnis das stärkste Band ist.

Aus diesen drei Verhältnissen konstruiert Tönnies ein weiteres, wichtiges Verhältnis: die Beziehung zwischen Vater und Kind (ebd.: 9 [1. Buch § 2]). Hier finden die anderen Verhältnisse ihre Einheit und Vollendung. Das Vatertum repräsentiert für Tönnies die Idee der Herrschaft im gemeinschaftlichen Sinne am reinsten: Herrschaft nicht als Gebrauch und Verfügung zum Nutzen des Herrschenden, sondern als Erziehung und Lehre als Vollendung der Erzeugung.

Im Zusammenleben wird sowohl die Arbeit als auch der Genuss in verschiedener Weise aufgeteilt.[4] Während im Verhältnis von Mann und Frau der Unterschied der natürlichen Kräfte ausschlaggebend ist, haben Geschwister gemeinsame Tätigkeiten. Ihr Verhältnis ist von Hilfeleistung, gegenseitiger Unterstützung und Förderung geprägt. Neben der Verschiedenheit des Geschlechts spielt hier die unterschiedliche mentale Begabung eine Rolle. Auf der einen Seite sammeln sich die eher geistigen und planenden Tätigkeiten (welche auch die Leitung umfasst) und auf der anderen die körperliche Arbeit und die Ausführung (eine Art Nachfolge und Gehorsam). Tönnies sieht dies als ein wechselseitiges Bestimmen und Dienen der Willen, welches sich immer in einem Kräftegleichgewicht befinden, im Sinne einer gegenseitigen Kompensation der Wirkungen aufeinander. Weiter ist er der Ansicht, dass die schwerere Arbeit, welche auch das Leiten anderer ist, einem größeren Genuss nach sich

[4] In der Gemeinschaft gibt es nach Tönnies keinen Tausch. Diesen begreift er als eine Art Handel, welcher der Hauptaspekt der Gesellschaft ist.

ziehe. Dieser Genuss könne aber auch ein Überlegenheitsgefühl sein, während der Geführte einen minderen Genuss habe, da er immer eine Unlust und eine Art Druck oder Zwang fühle – auch wenn Liebe, Gewohnheit und Dankbarkeit dies erleichterten. Tönnies beschreibt eine überlegene Kraft, die zum Wohle des Untergebenen oder seinem Willen gemäß ausgeübt wird, als Würde (Autorität). Er unterscheidet diese in drei Arten: die des Alters, die der Stärke und die der Weisheit. Diese drei Würden vereinigen sich wiederum in der Würde des Vaters.

Weiter unterscheidet er drei Ausformungen von Gemeinschaften, die aber auch vermischt vorkommen können: die des Blutes (Verwandtschaft), die des Ortes (Nachbarschaft) und die des Geistes (Freundschaft). Gemeinschaft des Geistes beruht nach Tönnies auf dem „bloßen Miteinander-Wirken und Walten in der gleichen Richtung, im gleichen Sinne“ (ebd.: 12 [1. Buch § 6]). Freundschaft wird am ehesten durch die Ähnlichkeit des Berufes oder die der Kunst gefördert. Doch muss sie durch leichte und häufige Vereinigungen geknüpft und erhalten werden. Hier zeigt sich die Bedeutung des Gedächtnisses. Es wirkt als Dankbarkeit und Treue. Das gegenseitige Vertrauen und der Glauben aneinander machen die Besonderheit solcher Beziehungen aus. Innerhalb der Freundschaft macht Tönnies die Würde des Meisters gegen Jünger, Schüler und Lehrlinge aus. Nach Tönnies bildet die geistige Freundschaft eine Art von unsichtbarer Ortschaft, die gleichsam durch eine künstlerische Institution, einem schöpferischen Willen lebendig ist (ebd.: 13 [1. Buch § 6]).

Tönnies sieht die Gemeinschaft als einen nicht zweckbegründeten, vorrationalen Zusammenschluss. Die Zusammengehörigkeit wird aus dem Wesenswillen, genauer dem Gefallen, der Gewohnheit und dem Gedächtnis abgeleitet. Gerade in den beiden letzten liegt auch die Erklärung für Gemeinschaften, die auf nicht familiäre Bindungen beruhen, beispielsweise innerhalb des Feudalsystems. Dies ist auf ideologische Konstruktionen, wie Nation, Ethnie und ähnliches übertragbar (Vgl. Elwert 1981: 441f.). Hier wird nachträglich versucht, eine Legitimationslücke durch eine scheinbar ursprüngliche Verbindung zu überbrücken. Dadurch gelten wieder Eigenschaften wie Vertrautheit, nicht individuelles zweckorientiertes Handeln etc. So sieht auch Craig Calhoun „community not as a variable structure of social relationships but as a form of common feeling, we encourage the notion that the commuity among neighbors and the community among citizens of the same nation are essentially similar“ (Calhoun 1991: 107). Er übernimmt den Begriff der vorgestellten Gemeinschaft von Benedict Anderson (Anderson 1983). Die Mitglieder vorgestellter Gemeinschaften müssen sich nicht persönlich kennen, dennoch sehen sie sich als Mitglieder einer Gemeinschaft, definiert durch gemeinsame, zugeschriebene Charakteristika, persönlichen Geschmack, Gewohnheiten oder Anliegen. Calhoun bezeichnet vorgestellte Gemeinschaften, wie beispielsweise Nationen, Geschlechter, ethnische Gruppen, Religionen als kategorische Identitäten (Calhoun 1991: 108).

Das wesentliche Charakteristikum einer Gemeinschaft ist also die Vorrationalität.

Gemeinschaftliche Bindungen sind nicht durch einen gemeinsamen Zweck zu begründen, sondern werden als natürlich begriffen. Im Falle nicht-verwandtschaftlicher Gemeinschaften wird diese Natürlichkeit durch die Konstruktion einer gemeinsamen Geschichte und ein Selbstverständnis als gemeinsame Kategorie hergestellt.

2.2 Soziale Gruppe

Laut Uwe Schimank besteht eine soziale Gruppe „aus einer Reihe von Individuen, die sich miteinander identifizieren und auf der Basis gemeinsamer Werte und Ziele in *informell* strukturierten Weisen interagieren“ (Schimank 2001: 200, meine Hervorhebung). Vier Eigenschaften sind für Gruppen charakteristisch: Interaktion, die Bildung einer Struktur, die Einigung auf Normen als Verhaltensrichtlinien und die Herausbildung eines Wir-Gefühls (ebd.: 201). Zunächst einmal müssen Personen miteinander in eine regelmäßige Interaktion treten, damit sie überhaupt eine Gruppe bilden können. Die Beteiligten müssen sich beachten und miteinander kommunizieren, um keine amorphe Masse zu bleiben. Ein einmaliges Zusammentreffen reicht noch nicht zur Gruppenbildung aus, sondern ist als eine Begegnung einzuordnen. Erst regelmäßiger Kontakt ermöglicht es, durch Interaktionen Strukturen herauszubilden. Dem einzelnen Mitglied wird innerhalb der Gruppe ein bestimmter Status zugewiesen und es übernimmt eine bestimmte Rolle. Anders als in formalen Organisationen sind die Rolle und die Statusposition nicht offiziell und unabhängig von der betreffenden Person angelegt worden, sondern werden informell entwickelt. Diese Entwicklung ist ein fortlaufender Prozess, da die Gruppenmitglieder sich in einem kontinuierlichen Aushandlungsprozess befinden (ebd.: 201f.).[5]

Neben dem regelmäßigen, strukturierenden Kontakt ist es für die Bildung einer Gruppe wichtig, sich auf gemeinsame Normen, Werte und Ziele zu einigen. Diese Normen, Werte und Ziele müssen nicht offen ausgesprochen werden, häufig existieren sie nur in einem stillen Einverständnis und werden für selbstverständlich gehalten (ebd.: 202). Die geteilten Normen sind für den inneren Zusammenhalt der Gruppe wichtig. Sie schreiben Handlungsweisen vor. Mit Hilfe von Normen wird ein Konformitätsdruck auf die Gruppenmitglieder ausgeübt. Allerdings ist die Normkonformität meist nur von kurzer Dauer, da Konsens und regelbasierte Konformität schnell zusammenbrechen. An ihre Stelle kann Kontrolle durch Gewalt treten.[6] Innerhalb einer Gruppe gibt es zwei Arten von Führung, die instrumentelle und die expressive Führung. Instrumentelle Führung ist bei der Leitung von Aufgabenausführung nötig. Expressive Führung betrifft die Pflege der Stimmung und der Beziehungen un-

[5]Dieser Prozess kann natürlich sehr langsam vor sich gehen. Als wichtig ist hier zu bedenken, dass die Strukturen in einer Gruppe nie wirklich fest sind. Im Vergleich zu einer formalen Organisation, bei der sich die formale Struktur erst mit einer offiziellen Änderung wandelt.

[6]Dies ist bei virtuellen Gruppen aufgrund der Nichtkörperlichkeit allerdings nicht möglich.

ter den Gruppenmitgliedern. Beide Führungstypen können in einer Person vereinigt sein, müssen es aber nicht (ebd.: 204).

Die letzte von Schimanks genannte Gruppeneigenschaft ist das Gefühl der Zusammengehörigkeit. Die Gruppenmitglieder sehen sich als eine Einheit (*Wir*), sie sind voneinander abhängig und unterscheiden sich von anderen Menschen (*Die*) (ebd.: 202). Das Gefühl der gemeinsamen Identität entsteht für Schimank also durch die Ab-/Ausgrenzung von anderen. Der Mitgliedschaftsgrad in Gruppen variiert, die Grenzen können sehr scharf gezogen sein – wer nicht eindeutig dazu gehört, ist ein Teil der Außenwelt. Oder aber der Grad der Mitgliedschaft verläuft fließend: es gibt einen Kern, die "große" Menge der Mitglieder, den Rand und die Mitläufer. Die Grenze der Inklusion verschiebt sich dann je nach Geschehen und Aufgabe. Zusätzlich können sich Gruppen auch in Untergruppen aufteilen (Neidhardt 1983: 20f.). Häufig werden die Gruppengrenzen für Außenstehende markiert, dies kann durch Symbole, eine spezielle Sprache, bestimmte Verhaltenskodices usw. geschehen. Eine weitere Möglichkeit zur Etablierung von Gruppengrenzen ist der Konflikt mit Außenstehenden. Der gemeinsame Feind schweißt die Gruppe zusammen und verstärkt das Wir-Gefühl (Schimank 2001: 206).

Gruppen beruhen meist auf direkten/unvermittelten Beziehungen. Manche Gruppen wachsen allerdings so weit an, dass sich nicht mehr alle Mitglieder persönlich kennen. In diesem Falle sind die Beziehungen organisatorisch überformt, so dass das Persönliche an Gewicht verliert.[7] Es können Mischformen zwischen Gruppe und Organisation entstehen (Neidhardt 1983: 15). Größere Gruppen verfügen aber auch über mehr Möglichkeiten der Arbeitsteilung, so ist für sehr komplexe Aufgabenstrukturen eine bestimmte Mindestanzahl von Beteiligten nötig (Schimank 2001: 203).

Friedhelm Neidhardt betrachtet die den persönlichen Charakter der Beziehungen in Gruppen näher (Neidhardt 1983: 14). Die einzelnen Gruppenmitglieder bringen ihre subjektiven Erfahrungen, Interessen und Gefühle mit in die Gruppe, die dort gezeigt, wahrgenommen, explizit thematisiert werden können und als Legitimation von Handlungen dienen (ebd.: 14). Dabei werden Gefühle laufend etikettiert, interpretiert und lizensiert, um sie geordnet zu verarbeiten. Dies passiert meist nicht explizit und sprachlich ausgedrückt, sondern bleibt oftmals latent und wird durch Körpersprache oder Betonung von Aussagen kommuniziert.[8] Innerhalb einer Gruppe kann sich so eine Gefühlskultur herausbilden, welche auch die nicht-verbale Ebene

[7] Dieter Claessens weist daraufhin, dass jede, noch so kleine Gruppe ihre Aktivitäten organisieren muss. Eine Organisation bildet sich erst durch die Verfestigung des Organisierens. Organisieren muss nach Claessens für jede Situation aktualisiert werden und beinhaltet eine verbale Abstimmung, wer was wann und wie tut, damit das gewünschte Resultat erzielt wird (Claessens 1983: 484ff.).

[8] In einer rein textbasierten Kommunikation ist dies (fast) unmöglich. Die einzige Abhilfe schaffen Betonungen durch die Schreibweise, beispielsweise werden im Internet Wörter, die nur aus Großbuchstaben bestehen, geschrien.

einschließt. Hier lassen sich die mehr oder weniger offen ausgehandelten Verhaltensregelungen finden, beispielsweise über was nicht gesprochen wird, was die idealisierte Vorstellung von Freundschaft ist, wer sich wann und warum schämen soll oder auch in was man großzügig zu sein hat (ebd.: 18).

Gruppenprozesse sind laut Neidhardt wesentlich durch Gefühle gesteuert. Dies bedeutet, dass die Entwicklung des weiteren Gruppengeschehens mehr von den einzelnen Gruppenmitglieder und ihren Gefühlen abhängt, als von Strukturen oder äußeren Einflüssen (ebd.: 17).[9] Gruppenprozesse ereignen sich als eine komplexe Abfolge persönlicher Zurechnungen des Handelns (ebd.: 17). Konsensbildung findet auf der Basis von Vertrauen statt und ist eine wichtige Voraussetzung, dass sich normalisierte Regelmäßigkeiten des Gruppenhandelns einspielen können, d.h. das eine Strukturierung des Handelns eintritt (ebd.: 20).

Das Besondere an dem Konzepts der sozialen Gruppe lässt sich in der Persönlichkeit der sozialen Beziehungen finden. Anders als bei einer Organisation – mit der ich mich gleich befassen werden – ist das Geschehen und die Struktur rein von den einzelnen Personen abhängig und nicht von Funktionsträgern. Trotz der Informalität sind Gruppen soziale Gebilde von dauerhaftem Bestand. Es bilden sich Strukturen und Normen heraus, die in einem dauerhaften Aushandlungsprozess unterliegen.

2.3 Organisation

Laut Uwe Schimank besteht „eine formale Organisation [...] aus einer Reihe von Individuen, deren Handlungen zur rationellen Erfüllung explizit formulierter Ziele präzise und zweckmäßig geplant sind“ (Schimank 2001: 200). Diese Ziele sollen dauerhaft erreicht werden, d. h. es sollen bestimmte Zustände erreicht werden, die nicht nur einmalig auftauchen, sondern dauerhaft sind oder kontinuierlich wiederkehren (Gukenbiehl 2002: 153).

Eine Organisation zeichnet sich dadurch aus, dass sie für diese Zielerreichung nach bestimmten Prinzipien strukturiert ist. Ihre Struktur ist rational geplant und allgemeinverbindlich (ebd.: 153). Sie schreibt vor, welche Einheit welche Aufgaben, Kompetenzen und Pflichten hat. Die Aufgabenteilung unter spezialisierten Einheiten geschieht unter dem Kriterium der Effizienz. Die in Arbeitsteilung erledigten Aufgaben werden durch innere Koordinierungsmechanismen zusammengeführt, also wieder funktional integriert (Schimank 2001: 210). Dies soll die Zielerreichung sichern und außerdem gewährleisten, dass die notwendige Kooperation innerhalb

[9] So kann sich beispielsweise eine Gruppe spalten, nur weil eine Person auf die andere eifersüchtig ist, oder eine Gruppe entsteht allein aus der gegenseitigen Sympathie der Beteiligten. Ebenso kann die Rekrutierung oder auch der Ausschluss von Mitgliedern durch rein persönliche Präferenz erfolgen.

der Organisation stattfindet. Hierzu müssen innerhalb einer Organisation bestimmte Kontrollfunktionen eingebaut werden. Dies wird durch die hierarchische Ordnung einer Organisation erreicht, bei der Vorgesetzte die Leistungen von Untergebenen kontrollieren.

Die Funktionsweise der Organisation, ihre Struktur, die Zielen, die formalisierten Abläufen, etc. sind in einem Regelwerk festgeschrieben. Dieses kann die Form von Gesetzen oder Satzungen haben, oder auch die einer mündlichen Vereinbarung (Gukenbiehl 2002: 154).

Die Struktur ist das Skelett einer Organisation. Die konkreten Menschen sind in ihm nur Einheiten, die Funktionen innerhalb der Organisation erfüllen. Zunächst wird die Funktion als solches geschaffen, welche in ein oder mehrere Ämter oder Stellen umgewandelt wird, erst dann wird ein Funktionsträger gesucht. Die Auswahl einer Person für eine bestimmte Stelle erfolgt nach deren Eignung die Aufgabe der Stelle meistern zu können. Die Person bindet sich mittels eines Vertrages an die Organisation.[10] Dieser Vertrag umschreibt gewöhnlich die Weise der Mitgliedschaft, die Aufgaben, Kompetenzen und Pflichten der Person der Organisation gegenüber (ebd.: 155). Sie setzt aber auch das Entgelt der Person für ihre Arbeit fest. Max Weber hebt immer wieder die in Bürokratien herrschende Trennung von Person und Amt, wie auch die Trennung von Amtsträgern und Amtsmittel hervor (Weber 1980: 124ff., 552f.).[11]

Neben der formalen Ordnung sind Organisationen aber auch von sozialen Prozessen geprägt. So finden innerhalb von Organisationen (fast) immer Gruppenbildungen statt (Schimank 2001: 215) – dies soll hier nicht von Interesse sein, da dem Thema Gruppen ein eigener Abschnitt gewidmet wurde. Auch wenn Organisationen wie oben erwähnt durch Planung konstruiert und verändert werden, entstehen sie nicht aus dem Nichts. So können sich Personen oder auch Gruppen auf bestimmte organisatorische Strukturen einigen, wenn sie sie für die Erledigung bestimmter Aufgaben nutzen wollen. Dabei wird oft an eine bestehendes institutionelles Umfeld angeschlossen (vgl. Claessens 1983: 468ff.).

Organisationen haben den Charakter von Instrumenten. Ihr Sinn liegt in der Erreichung von vorher festgelegten Zielen. Hierzu bilden sie eine effektivitätsorientierte Struktur aus. In dieser Struktur treten die Personen hinter den Funktionen, die sie erfüllen, zurück und sind nur noch als Funktionsträger sichtbar. Die Stärke von Organisationen im Vergleich zu anderen Arten des Zusammenschlusses liegt in der Bewältigung von bestimmten Aufgaben.

[10] Der Vertrag muss keine schriftliche Form haben. Bindet sich eine Person an eine Organisation, ohne eine schriftliche Abmachung, wird der Vertrag durch die faktische Mitarbeit innerhalb der Organisation geschlossen.

[11] Auch wenn Weber immer von Bürokratien spricht, lassen sich viele seiner Aussagen auch auf Organisationen im weiteren Sinne übertragen.

2.4 Netzwerk

Netzwerke beschreiben modellhaft die Struktur von Verbindungen zwischen einzelnen Personen oder auch ganzen Gruppen. Was jeweils als ein Netzwerk angesehen wird, bestimmt sich durch eine künstliche Grenzziehung bei der Festlegung des Untersuchungsfeldes. Die Untersuchung kann auf verschiedene Weise gestaltet werden. Zum einen kann eine bestimmte Population auf ihre Strukturen hin untersucht werden, andererseits gibt es die Möglichkeit, die Kontakte einer Person zu verfolgen. Anders als die Untersuchung von Gemeinschaften, sozialen Gruppen und Organisationen richtet die Untersuchung von Netzwerkstrukturen ihr Augenmerk auch auf jene Verbindungen, die über Grenzen der Zusammengehörigkeit hinaus gehen (Wellman 2003: 127).

Für die Analyse von Netzwerken beschreibt Barry Wellman sechs Merkmale: Dichte, Abgrenzung, Reichweite, Ausschließlichkeit, soziale Kontrolle und die Stärke der Bindungen (Wellman 2003). Dichte fragt nach dem Grad der Verbundenheit der Netzwerkmitglieder untereinander, der von losen Gruppen mit selektiver Kommunikation bis zu engen Gruppen mit regem allseitigen Kontakt reicht (ebd.: 132ff.s).

Abgrenzung bezieht sich auf den Anteil der Bindungen, die innerhalb des Netzwerkes liegen. Stark abgegrenzte Netzwerke zeichnen sich durch (fast) ausschließlich interne Kontakte aus. Die verbleibenden Außenkontakte werden durch wenige *Pförtner* gepflegt. Mitglieder von lose abgegrenzten Netzwerken haben viele Bindungen nach außen (ebd.: 137). Während also Dichte nach der inneren Gestaltung eines Netzwerkes fragt, beschäftigt sich Abgrenzung mit den Außenkontakten.[12]

Die Reichweite bezieht sich auf die Größe und Heterogenität der Netzwerkpopulation. Netzwerke mit großer Reichweite haben häufig keine gleichmäßige Verteilung der Bindungen. Die Beziehungen nehmen die Gestalt von Clustern an, welche über wenige Brücken (Pförtner) verbunden sind (ebd.: 140f.).

Ausschließlichkeit beschreibt die Zugänglichkeit von Beziehungen für Außenstehende. Auf der einen Seite der Skala bleiben die Interaktionen in einer bilateralen Beziehung, auf der anderen Seite stehen sie allen Außenstehenden zur Verfügung (ebd.: 143).

Die Frage nach der sozialen Kontrolle innerhalb eines Netzwerkes beschäftigt sich damit, ob die Bindungen einer Person durch äußere Faktoren gefördert, begrenzt oder kontrolliert werden. In dichten, abgegrenzten Netzwerken wird meistens durch Gruppenzwang, Führungspersönlichkeiten u. ä. die Einhaltung normativen Sozialverhaltens kontrolliert. In losen Konstellationen ist soziale Kontrolle kaum möglich, da hier die Möglichkeit des Ausweichens viel größer ist. Kontrolle findet hier weniger

[12] In der praktischen Analyse ist das Konzept der Abgrenzung aber nur sehr schwer anzuwenden, da die Definition, was das Netzwerk eigentlich umfasst oft nicht klar genug gemacht werden kann.

durch Zwang als durch internalisierte Normen und standardisierte organisatorische Verfahren statt (ebd.: 144f.).

Wellman beschreibt die Stärke der Bindungen als ein mehrdimensionales Konstrukt, dessen Variablen in der Regel miteinander korrelieren (ebd.: 147). Diese Variablen sind: soziale Nähe, Freiwilligkeit, Multiplexität (Bandbreite) und Kontakthäufigkeit.[13] Das Konzept der Stärke der Bindungen geht auf Mark Granovetter zurück (Granovetter 1973). Starke Bindungen haben eine größere Bandbreite der inhaltlichen Verbindungspunkte. Sie zeichnen sich durch emotionale Unterstützung, den Austausch von Waren und Dienstleistungen, Geselligkeit und Zusammengehörigkeitsgefühl aus. Schwachen Bindungen bringen Personen zusammen, die sich gesellschaftlicher viel stärker unterscheiden. Dadurch ermöglichen sie einen Zugang zu anderen sozialen Sphären und anderen Ressourcen (Wellman 2003: 147). Beziehungen in dichten, abgegrenzten Netzwerken sind häufig unfreiwillig, aber auf der anderen Seite haben sie einen höheren Grad an sozialer Intimität. In losen Netzwerken werden dagegen die Beziehungen freiwillig eingegangen. Nach Wellman beruhen diese Kontakte auf gemeinsamen Interesse und gegenseitigen Vorteilen (ebd. 148). Allerdings ist hier zu bemerken, dass durch die Freiwilligkeit der Bindung die Kosten für die Aufrechterhaltung höher sind, als bei dichten Netzwerken. Die Stärke der Bindungen befasst sich also mit der Qualität der Beziehungen.

Die Analyse von Netzwerken kann helfen, zunächst einmal die Verbindungen von Personen und Gruppen aufzeigen. Sie muss aber auch die Qualität der Beziehungen mitberücksichtigen, um aussagefähig zu sein. Hierbei sollte auch auf den Unterschied zwischen direkten und indirekten (vermittelten) Beziehungen geachtet werden. Durch die Fokussierung auf die Bindungen erscheint die Netzwerkanalyse besonders für das Studium informell strukturierter Formationen geeignet. Allerdings müssen formale Regeln und Verfahren auch in der Analyse mit beachtet werden, da sie die Funktion der Bindungen (z. B. Kontrolle) nachhaltig beeinflussen.

2.5 Soziale Bewegung

Nach Rucht/Neidhardt stellen soziale Bewegungen „soziale Gebilde aus miteinander vernetzten Personen, Gruppen und Organisationen dar, die mit kollektiven Aktionen Protest ausdrücken, um soziale bzw. politische Verhältnisse zu verändern oder um sich vollziehenden Veränderungen entgegenzuwirken“ (Rucht/Neidhardt 2001: 540). Soziale Bewegungen sind keine durchorganisierten Gebilde. Häufig schließen sie aber Organisationen ein, auch wenn sie selbst keine sind. Sie haben eher die Gestalt von

[13] Untersuchungen von Wellman und anderen ergaben, dass die Häufigkeit ein nur bedingt geeignetes Kriterium für die Bindungsstärke ist, daher wird es in der folgenden Ausführung nicht weiter beachtet (Wellman 2003: 147).

Netzwerken. So haben sie keine klare Mitgliedschaftszuschreibung, die von der Umwelt eindeutig abgrenzbar ist. Aktivisten und Teilnehmer gehören zu der Bewegung, Unterstützer und Sympathisanten dagegen werden zur Umwelt gerechnet. Innerhalb von sozialen Bewegungen gibt es keine flächendeckende Durchorganisierung oder eindeutige Differenzierung von Positionen und Rollen. Die beobachtbaren Verhaltensunterschiede der Akteure sind eher durch eine unterschiedlich starke individuelle Motivation und spontane Einsätze ungleich verteilter Talente begründet. Das heißt, dass bei sozialen Bewegungen, anders als bei Organisationen, sich Positionen nur schwer von den spezifischen Menschen trennen lassen. Entscheidungsstrukturen sind entweder durch eine charismatische Herrschaft bestimmt, oder sie weisen eine hohe Dezentralität auf. Der Umstand, dass sich die Handlungsfähigkeit sozialer Bewegungen allein aus dem inneren Engagement ihrer Mitglieder ableitet, fördert die mangelnde Hierarchisierbarkeit. Als Bindemittel ist die Verbindung von individuellem Motiv und kollektivem Zweck entscheidend. So müssen sich soziale Bewegungen in Ermangelung anderer Ressourcen nach den Wünschen der zu motivierenden Mitglieder richten, um deren inneres Commitment und darauf folgend deren Engagement zu erlangen. Nach Rucht/Neidhardt bildet sich „in der commitment verlangenden und zugleich bekräftigenden Handlungspraxis [...] ein Wir-Gefühl, [...] kollektive Identität heraus, die eine soziale Bewegung von einem reinem Zweckverband [...] unterscheidet (ebd. 541).

Soziale Bewegungen sind wie schon oben erwähnt keine Organisationen, sie haben eher die Gestalt von Netzwerken. Man könnte sie als mobilisierte Netzwerke von Netzwerken beschreiben. Aufgrund des geringen Organisationsgrades und ihrer diffusen Entscheidungsstrukturen sind ihre Steuerungskapazitäten eher gering. Sie eignen sich mehr zum Anstoßen oder Blockieren als zum planvollen Durchsetzen von Zielen.

Es gibt verschiedenen theoretischer Ansätze zur Erklärung und zur Analyse von sozialen Bewegungen. Nach der Deprivatonsthese sind Akteure aufgrund ihres Elends und ihrer Not bereit, sich zusammen zu tun und sich zu wehren (ebd.: 550). Die relative Deprivationstheorie besagt, dass die Betroffenen ihre objektive Situation erst in Abhängigkeit von ihren Ansprüchen als schlecht wahrnehmen. Die Differenz der Wahrnehmung zwischen dem was man beansprucht und das was man hat führt zu Frustration. Diese kann also nicht nur durch eine Verschlechterung der Lebenssituation, sondern auch durch eine Steigerung der Erwartungen entstehen und gegebenenfalls zum Protest führen (ebd.: 551).

Die Analyse von Frames (Rahmen im Sinne des Deutungsrahmens) stellt heraus, dass soziales, so auch kollektives Handeln durch Sinnkonstruktionen der Akteure bestimmt wird. Bei der Analyse von Frames ist zwischen drei Dimensionen oder Subframes zu unterscheiden: Der *Diagnostic Frame* befasst sich mit der Definition des Übels; Der *Identity Frame* definiert ein „Wir“ gegenüber einem „Sie“, das sich

durch andere Interessen oder Werte auszeichnet – so wird das Gemeinschaftsgefühl der Betroffenen geschaffen;[14] Der *Prognostic* oder *Agency Frame* beschäftigt sich mit den Strategien, Taktiken und Zielen (ebd.: 551).

Ein weiterer Ansatz untersucht die mobilisierten Ressourcen. Es wird dabei davon ausgegangen, dass aufgrund der gegenwärtigen sozialen Probleme immer genügend Unzufriedenheit und Mobilisierungsbereitschaft vorhanden sind. Die Frage ist nur, ob es für die jeweilige Problemlage Bewegungsunternehmer gibt, die es schaffen, Bewegungsorganisationen aufzubauen und durch sie Anhänger und Unterstützer zu mobilisieren. Der *Resource Mobilization* Ansatz hat die Annahme, dass Bewegungsaktionen strategisch von quasi-professionalisierten Bewegungsmanagern zweckrational geplant und gestaltet werden (ebd.: 552f.).

Der letzte hier vorgestellte Ansatz stellt die Umwelt der sozialen Bewegungen in den Mittelpunkt. Zunächst werden die relevanten Bezugsgruppen festgestellt. Auf diese Gruppen, Organisationen und Institutionen bezieht sich die Bewegungsorientierung primär, da sie für die Erfolgschancen der Bewegung als besonders entscheidend erscheinen. Die Konzepte der Gelegenheitsstrukturen und des politischen Prozesses untersuchen die Stabilisierungsmöglichkeiten und Erfolgschancen von sozialen Bewegungen, die sich im Handlungsfeld in Abhängigkeit von deren Bezugsgruppenkonstellation ergeben. Die Parameter für die Gelegenheitsstruktur sind das Ausmaß an Offenheit eines politischen Systems, dies wird beeinflusst durch den Grad seiner Demokratisierung, aber auch durch das Ausmaß der Dezentralisierung und die Stabilität der politischen Strukturen. Die These ist, dass die Handlungschancen einer sozialen Bewegung wachsen, je differenzierter und dezentraler die Bezugsgruppenkonstellation seiner Umwelt ist. So kann die soziale Bewegung eher Ansatzpunkte für die zum Erreichen der Ziele notwendige Unterstützung durch die Bezugsgruppen finden (ebd.: 554).

Der Ansatz der sozialen Bewegungen zielt stark auf politische Felder ab, was hier entsprechend angepasst werden muss: Die Zielsetzung des Veränderns bezieht sich im Bereich der Freien Software auf Marktmechanismen. Zu untersuchen wäre, inwieweit diese Veränderungsmotivationen die Formationen der Entwickler tatsächlich beeinflusst. Die Frage der Gegendefinition und der zugehörigen Frames erscheint dabei als besonders relevant.

2.6 Subkultur

Fritz Sack sieht in der „Subkulturtheorie“ eher eine Perspektive, als ein Aussagensystem mit theoretischer Geschlossenheit (Sack 1971: 262). Nach ihm lässt sich die

[14] Da der Zusammenhalt und der Handlungsbedarf durch die Ausarbeitung eines Feindbildes geschaffen wird, sprich man auch von *Motivational Frame*.

Attraktivität des Konzepts Subkultur auf die Eigenschaften und Funktionen des Mutterbegriffes Kultur zurückführen. Dessen wissenschaftliche Bedeutung liegt in dem generellen Rahmen der Betrachtung und Interpretation für die Beschreibung und Analyse menschlichen Verhaltens in den verschiedenen Kulturen. Sack bezieht sich auf Ward Hunt Goodenoughs Auffassung von Kultur als ein Instrument zur Organisation der Erfahrungen, die von einer Gesellschaft geteilt werden (ebd.: 265; vgl. Goodenough 1963: 259). Hiermit sind alle Standards der Wahrnehmung, des Urteilens, des Bewertens, der sprachlichen Aufbereitung der Umwelt und des Handelns gemeint, die einem Individuum in seiner Gesellschaft kulturell vorgegeben sind. Diese Standards werden in der Regelhaftigkeit menschlichen Verhaltens, im Wiederholen von typischen Handlungsabläufen sichtbar, sie manifestieren sich als Regeln, Muster, Standards, Normen, Erwartungen, Verhaltensvorschriften. Nach Sack organisieren sich die einzelnen Komponenten der Kultur zu komplexen Systemen der Moral, der Ethik, der Ästhetik, des Rechts, der Religion und bilden hierarchisch geordnete und in sich geschichtete Wertesysteme aus.

Sack weist darauf hin, dass nicht alles menschliche Verhalten kulturell bedingt ist (ebd.: 266). Neben kulturell beeinflussten Verhalten gibt es auch Handlungen, die situationsgesteuert sind, im Sinne der Notwendigkeit der Anpassung an externe Umweltbedingungen, die außerhalb der Kontrolle der Handelnden liegen. Der Ursprung von kulturell geprägtem Verhalten lässt sich alleinig in durch tradierte Normen vorgegebenen Verhaltensregeln finden, situationsbedingtes Verhalten ist dagegen an die spezifische Situation gebunden, in der es auftritt. Die Grenze zwischen den beiden Verhaltensformen ist nach Sack aber durchlässig und daher nicht genau bestimmbar (ebd.: 266).

Diese Charakteristika der Kultur gelten auch für Subkulturen. Eine Subkultur ist – wie das Wort es schon besagt – eine *unter*geordnete Kultur einer größeren Gesamtkultur. Nicht alle Personen messen den Normen, Werten und Symbolen der Gesamtkultur die gleiche Geltung und Bedeutung bei (Peuckert 2002: 114). Subkulturen sind das Resultat von Interaktionen zwischen Menschen, die gleiche oder ähnliche Problemen mit der Identifizierung mit dem dominierenden Norm- und Wertsystem haben und es daher vernünftiger finden, nach anderen Normen zu handeln, um eben die sich dann die Subkulturen bilden. J. Milton Yinger weist darauf hin, dass es innerhalb einer (Gesamt-)Kultur eine an soziale Rollen gebundene Normdifferenzierung gibt (Yinger 1960: 627). Der Unterschied zu einer Subkultur besteht darin, dass die Rechte und Pflichte einer sozialen Rolle bekannt sind und von allen anderen in der betreffenden Gesellschaft akzeptiert werden – und damit Teil der Kultur bleiben. Subkulturen dagegen beziehen sich auf Normen, die sie von der Gesamtgesellschaft trennen und nicht darin integrieren. Denn die Normensysteme der Subkulturen sind für die anderen Mitglieder der Gesellschaft unbekannt, unzugänglich, auf sie wird hinabgesehen oder sie werden als spaltend für die Gesellschaft betrachtet (ebd.:

627f.).

Eine Ausdifferenzierung des Konzepts der Subkultur ist die Kontrakultur. Das Normensystem jeder Subkultur weicht von dem der Gesamtgesellschaft ab. Das einer Kontrakultur steht im direkten Gegensatz zu dem der Gesamtkultur – der Konflikt mit der Gesamtkultur ist das zentrale Element der Kontrakultur (ebd.: 629).

Problematisch an der wissenschaftlichen Verwendung des Begriffes der Subkultur ist, dass er häufig in Verbindung mit kriminellen Milieus oder auch Teilen der Unterschicht gebracht wird, was die Theoriebildung stark beeinflusst hat (vgl. Peukert 2002: 104ff.; Girtler 1995: 26ff.; Girtler 1991: 385f.; Sack 1971: 270ff.; Yinger 1960: 631ff.). Einen etwas anderen Ansatz verfolgt Roland Girtler, nach ihm ist die Gesellschaft nichts Einheitliches, sondern sie ist „multikulturell“ (Girtler 1995: 15), d.h. sie besteht aus einer Vielzahl von Gruppen mit jeweils eigener Kultur, die durch spezielle Symbole, Geheimsprachen, bestimmte Kleidung oder auch Rituale sichtbar werden kann (ebd.: 21). Dennoch wird eine Gesellschaft als Ganzes von einer übergreifenden Kultur integriert, die sich insbesondere durch eine gemeinsame Sprache und durch gemeinsame Grundnormen auszeichnet (ebd.: 29). Subkulturen gründen ihr Normensystem meist nur auf wenige Werte, welche sich auf einen bestimmten Ausschnitt des Lebens konzentrieren. Nur in diesen sind sie für ihre Mitglieder handlungsleitend, deren sonstiges Handeln orientiert sich weiter an dem Normensystem der Gesamtkultur. Subkulturen integrieren ihre Mitglieder also nur punktuell. Subkulturelles Verhalten wird durch Kontakte und Kommunikation in der betreffenden Gruppe erlernt und ist nicht in der Sozialstruktur festgelegt (ebd.: 27), d.h. die Integration in eine Subkultur findet über Enkulturation statt und wird durch bestimmte Lebensstile aufrecht erhalten.

Subkultur dient, wie Kultur auch, der Integration einer Gruppe. Sie regelt das gemeinsame Leben und Handeln, indem sie Wahrnehmungs- und Bewertungsrahmen vorgibt. Kultur wird durch Rituale, Lebensstile, usw. produziert und aufrecht erhalten. Subkultur unterscheidet sich von der Gesamtkultur durch eine partiell andere Wertehierarchie und daraus folgend durch ein anderes Normensystem. Im Falle einer Kontrakultur sind die Normen nicht nur einfach verschieden von denen der gesamtgesellschaftlichen Kultur, sondern stehen im direkten Konflikt zu ihnen.

2.7 Soziale Welt

Der Ansatz der sozialen Welt entwickelt sich aus dem (symbolischen) Interaktionismus (Strübing 2002). Die hier vorgestellte Ausarbeitung stammt von Anselm Strauss.

Eine soziale Welt bildet sich nach Strauss um eine Kernaktivität. Bei diesen Handlungen kann es sich um Tätigkeiten wie Angeln, Forschen, kein Fleisch Essen, usw.

handeln. Soziale Welten können in ihrer Größe stark variieren, ebenso in der Anzahl und den Varianten der Kernaktivität, der organisatorischen Komplexität, dem technischen Entwicklungsstand, der ideologischen Ausprägung, der geographischen Verbreitung, usw. (Strauss 1982: 172).

Soziale Welten sind wiederum in soziale Subwelten unterteilt, die ihrerseits wieder in weitere Subwelten unterteilt sein können. Strauss arbeitet drei mögliche Weisen der Entstehung von Subwelten heraus:[15] 1. Eine soziale Subwelt kann aus einer anderen entstehen, aufgrund einer Spezialisierung der Kernaktivität, einer räumlichen Veränderung oder einem Wandel der Technologie; 2. Es kann in einer Welt zu so großen Differenzen zwischen verschiedenen Segmenten kommen, dass die neue soziale Subwelt mehr einer Abspaltung gleicht als einer Ausdifferenzierung; 3. Soziale Subwelten können durch die Kreuzung von verschiedenen sozialen Welten entstehen.

Zur Bildung einer sozialen (Sub-)Welt reicht die Kernaktivität alleine aber noch nicht aus. Die Individuen müssen nicht nur erkennen, dass sie und andere die gleiche Tätigkeit ausführen, sondern es muss eine gemeinsame Definition entstehen, welche bestimmten Tätigkeiten es Wert sind getan zu werden. Zusätzlich muss beim einzelnen das Bewusstsein vorhanden sein, diese bestimmten Tätigkeiten *selbst* auszuüben (ebd.: 174). Hieraus folgt auch, dass soziale Welten erst durch handlungspraktische Konsequenzen sichtbar werden (Strübing 2002: 187). Der Wert einer Tätigkeit kann sich beispielsweise durch Spaß, Nützlichkeit, ästhetisches Recht, Moral oder Wahrheitsfindung begründen (Strauss 1982: 174; Strauss 1984: 128). Die neue Subwelt steht mit der alten, elterlichen sozialen Welt oder auch parallelen Subwelten um Ressourcen (Raum, Geld, Ausrüstung, Zugang zur Öffentlichkeit) und Legitimität in Konkurrenz. Sie muss ihre Tätigkeiten, Ideen, Technologien, Organisationen, etc. gegenüber sich selbst, aber auch nach außen hin legitimieren (Strauss 1982: 175). Wichtig ist hierbei, dass sie sich insbesondere nach außen klar von anderen sozialen Welten unterscheidet.

Einige sozialen Welten schreiben die Geschichte der Ursprungswelt neu. Deren Geschichte wird neu ausgewertet und interpretiert: wichtige Persönlichkeiten werden degradiert, vorher nebensächliche werden äußert wichtig. So kommt es, dass die lange „übersehenen", „wahren" Gründerväter endlich „entdeckt" werden, zu denen eine Abstammung beansprucht wird. Ähnlich ist es mit historischen Ären, manche werden für irrelevant erklärt, andere bekommen plötzlich Gewicht verliehen. Einige der neuen Ären oder Ahnen werden von fremden sozialen Welten herangezogen. Der Beginn der Geschichtsneuschreibung geschieht durch Diskussionen und spontane Entdeckungen durch die Mitgliedern der entstehenden Welt. Die neue Welt konsolidiert sich und entwickelt einen Apparat, der das Studieren, den Zugang und das Weitergeben der „wahren" Geschichte, sowie die Verbreitung von Neuigkeiten

[15] Seine Formulierung läßt darauf schließen, dass er weitere Entstehungsarten nicht ausschließt (Strauss 1982 :174).

ermöglicht. Gleichzeitig wird die gegenwärtige Geschichte geschrieben und die intra- und inter-Welt Geschehnisse interpretiert (Strauss 1984: 130f.).

Die Räume, in denen debattiert, verhandelt, gekämpft und manipuliert wird bezeichnet Strauss als Arenen (Strauss 1978: 124). Hier begegnen sich auch verschiedene soziale Welten und Subwelten, um über Themen zu streiten, die als wichtig für die eigene Zukunft wahrgenommen werden. Für neu entstandene Subwelten liegen wichtige Arenen in der Ursprungswelt. Es wird dort über die Verteilung von Ressourcen und den Zugang zu einer breiteren Öffentlichkeit verhandelt. Die kleineren Arenen in der Subwelt selbst beschäftigen sich eher mit den internen Beziehungen. Die Frage nach Authentizität ist hierbei ein wichtiges Thema. Die Themen der internen Arenen können von außen in die Subwelt kommen, durch die Ursprungswelt oder auch von ihr unabhängig. So kann die Subwelt in in größere Arenen hineingezogen werden. Eine Folge von internen Arenen, deren Diskussionspunkte nicht gütlich gelöst werden können, ist eine weitere Abspaltung einer so entstehenden Subsubwelt. Der Prozess der Segmentierung geht dann für diese von vorne los.

Ein wichtiges Element zur Ordnung oder Strukturierung innerhalb von sozialen Welten ist die Authentizität bezüglich der Kernaktivität. Jedes Mitglied einer sozialen Welt wird mit der Kernaktivität verbunden, allerdings werden einige als authentischer als andere angesehen, d.h. sie sind stärker repräsentativ. Authentizität betrifft die Qualität der Handlungen wie auch die Beurteilung, welche Aktivitäten essentieller sind als andere (Strauss 1978: 123). In neuen Welten entstehen schnell Standards zur Beurteilung von Tätigkeiten, zunächst implizit, später explizit und möglicherweise auch formalisiert. Dieser Prozess wird durch Weitergabe im Sinne von Lehren sehr gefördert. So werden die zentralen Figuren der betreffenden Welt und die Art ihrer Handlungsausführung genau beobachtet und beeinflussen so sie Standards. Einige Mitglieder der sozialen Welt erwerben das Recht der legitimen Beurteilung über Tätigkeiten und Produkte, sie verleihen Preise und geben Zertifikate aus. Aber auch die legitimierten Beurteiler können sich in ihrer Entscheidung irren oder auch die Standards vorsätzlich hintergehen. Daher brauchen soziale Welten Personen, die ihrerseits die Beurteiler kontrollieren. Strauss schlägt zur Untersuchung vor, nachzuverfolgen, wer die Macht hat zu entscheiden, was als authentisch gilt, wer darüber entscheidet, welche Mitglieder authentischer sind, und aufgrund welcher sozialen Mechanismen dies festgelegt wird (Strauss 1978: 123).

Das Besondere am Konzept der sozialen Welten ist, dass es den Zusammenschluss von Menschen um eine bestimmte Aktivität herum betrachtet. Die Strukturierung innerhalb diese Formation findet über den Grad dessen, was Strauss als Authentizität bezeichnet, statt. Hierdurch bietet dieser Ansatz eine Möglichkeit zu ermitteln, wie sich Hierarchien innerhalb von Formationen herausbilden. Außerdem versucht er, die Beziehungen zwischen verschiedenen Zusammenschlüssen genauer zu beschreiben.

2.8 Zusammenfassung

Die hier vorgestellten Modelle sozialer Formationen haben alle ihre eigene Perspektive, auch wenn sie bei einigen Aspekten zu sich überschneidenden Aussagen kommen. Im folgenden sollen zunächst noch einmal die wichtigsten Punkte wiederholt werden, die für die Analyse interessant und förderlich erscheinen.

Das Konzept der Gemeinschaft (insbesondere in der Bearbeitung von Ferdinand Tönnies) betont die nicht-zweckgebundene, vor rationale Bindung zwischen Menschen. Sie beruht auf Gefallen, Gewöhnung und Gedächtnis, je nach Art des Verhältnisses. Für diese Arbeit dürfte das Verhältnis der Freundschaft am interessantesten sein, da es häufig auf geteilten Interessen an bestimmten Themen beruht. Ein erweiterter Ansatz von Gemeinschaft findet die natürliche Zusammengehörigkeit in der Konstruktion und Betonung der kollektiven (Abstammungs-)Geschichte. Für die Untersuchung des Feldes der Entwickler Freier Software stellen sich daher die Fragen, wie das Feld entstanden ist, welche interne Geschichtsschreibung es gibt und ob diese eine (quasi-)natürliche Zusammengehörigkeit zu erschaffen versucht. Hierbei ist die Überprüfung, ob die Bindungen als zweckfrei verstanden werden von entscheidender Bedeutung.

Die Theorie sozialer Gruppen und Organisationen beschäftigen sich mit der inneren strukturellen Gestaltung von sozialen Formationen. Soziale Gruppen zeichnen sich durch die Persönlichkeit der Beziehungen aus. Durch Interaktionen bilden sich die informelle Strukturen einer Gruppe aus. Jedem Gruppenmitglied wird eine Rolle zugewiesen und es erwirbt sich einen gewissen Status. Eine besondere Rolle ist die des Gruppenführer, die es in zweifacher Ausprägung gibt: der instrumentelle Führer, der organisatorische Aufgaben übernimmt, und der expressive, der für den Gruppenzusammenhalt sorgt. Neben geteilten Normen, Werten und Zielen wird die Stabilität von Gruppen auch durch ein Wir-Gefühl gestärkt. Diese entsteht insbesondere durch die Abgrenzung nach außen, was durch den Aufbau von Feindbildern gefördert werden kann. Wie bereits erwähnt sind Gruppen durch Persönlichkeit und Informalität charakterisiert, so sind die Strukturen und auch die Werte und Normen innerhalb von Gruppen niemals wirklich fest, sondern befinden sich immer in einem Aushandlungsprozess. Das Konzept der sozialen Gruppe soll helfen das Zusammenagieren der Entwickler in den einzelnen Projekten zu analysieren. Hierbei ist interessant, wie Projekte entstehen, wie sich ihre Strukturen herausbilden, wie diese sich wandeln, welche Rollen ausgebildet und wie diese zugewiesen werden, wer zum Projekt in welchem Maße dazu gehört, welche Normen und Werte vorherrschen und wie die Abgrenzung nach außen gestaltet ist.

Organisationen sind Instrumente für eine effiziente Zielerreichung. Zunächst werden die Aufgaben erörtert, die gelöst werden sollen, erst danach beginnt die Planung wie dies genauer geschehen soll. Hierzu wird eine formale Struktur geplant, in welcher

Menschen nur als Funktionsträger wahrgenommen werden, die einem vorab festgelegten Kompetenzbereich haben. Wichtig ist hierbei die Trennung von Person und Aufgabe. Die Betrachtung der organisatorischen Strukturen der Projekte beschäftigt sich mit den Fragen, ob es eine formalisierte Organisation überhaupt gibt, wie diese ausgestaltet ist, welcher Mittel sie sich bedient und welche Art von Zielen durch die Organisation zu erfüllen versucht wird. Dabei sollen auch Prozesse der Organisationsbildung berücksichtigt werden.

Die Analyse von Netzwerken konzentriert sich auf die Beobachtung von existenten Verbindungen zwischen einzelnen Menschen oder auch zwischen Organisationen. Hier soll sie sowohl die Untersuchung von einzelnen Projekten, aber auch die des ganzen Feldes unterstützen. Dies soll unter anderem helfen, die Frage nach dem Grad der Fragmentierung des Feldes zu beantworten.

Soziale Bewegungen sind amorphe Gebilde, die sich mit dem Ziel zusammenschließen, gezielt Wandel einzuleiten oder stattfinde Veränderungen aufzuhalten. So ist Motivation das entscheidende Charakteristikum von sozialen Bewegungen. Falls das Phänomen der Entwickler Freier Software sich als eine soziale Bewegung herausstellen sollte, muss der Wille zur Veränderung oder zur Verteidigung des Status Quo als *primäre* Motivation vorhanden sein. Dies gilt es durch die Untersuchung der Motivation der Beteiligten zu klären. Zusätzlich sollte dann die Bewegung auf ihre Komponenten, Strategien und Ziele untersucht werden.

Eine (Sub-)Kultur dient zunächst einmal der Integration einer Gruppe, indem sie das gemeinsame Leben und Handeln mit vorgegebenen Wahrnehmungs- und Bewertungsrahmen regelt. Subkulturen bilden Werte heraus, die von denen der Gesamtkultur partiell abweichen. Die Werte schlagen sich in einem Normensystem nieder, das sich durch Lebensweisen, Rituale und Diskurse manifestiert. Diese kulturellen Prozesse gewährleisten die Produktion und Erhaltung der (Sub-)Kultur. Es soll also in der Analyse nach den Werten und Normen des Feldes gefragt werden. Ob man von einer Subkultur der Entwickler Freier Software sprechen kann, ist an die Frage gekoppelt, ob diese Werte und Normen großflächig geteilt werden und ob sie zu einer Kulturproduktion führen, die das Handeln anleitet und begründet.

Der zentrale Aspekt bei sozialen Welten (nach Anselm Strauss) ist die Kernaktivität, um die sich die soziale Welt gruppiert. Der Ansatz orientiert sich also zunächst mehr an Handlungen, als an einer feststehenden Struktur. Das strukturierende Element beim Konzept der sozialen Welten ist die Authentizität. Zum einem wird sie zur Abgrenzung von anderen sozialen (Sub-)Welten benutzt, andererseits legitimiert sie die Hierarchiebildung innerhalb einer sozialen Welt. Die Betrachtung des Phänomens mit der Perspektive der sozialen Welten hat ihren Reiz in der Betonung der Tätigkeit, die die Beteiligten verbindet. Das Konzept der Authentizität bietet eine Hilfe zur Analyse, welche Qualitätsstandards bei den Aktivitäten existieren und wie diese sich entwickeln. Schließlich stellt der Ansatz der sozialen Welten noch einige interessante

Anregungen zur Entstehung und Entwicklung des Feldes bereit.

Kapitel 3

Die Geschichte

Freie Software entstand in den 1980er Jahren. Um das Phänomen allerdings vernünftig einordnen und analysieren zu können, muss man ein bis zwei Jahrzehnte in der Geschichte zurückgehen, in die 60er und 70er Jahren des 20. Jahrhunderts. Zur dieser Zeit prägten sich einige Arbeitsweisen und ethische Konzepte heraus, die noch heute im Bereich Freier Software zu finden sind. Die folgende Beschreibung der historischen Ereignisse soll diese Punkte herausarbeiten. Bei der Schilderung der Ereignisse sollen auch die Strukturen von Kooperation, Motivationsmuster und die Umstände, die zu den jeweiligen Entwicklungen führen, aufgezeigt werden.

3.1 Die Ursprünge

In den 1950er Jahren bedeutete Computer immer Großrechner. Diese waren ausschließlich in Universitäten, Forschungseinrichtungen, große Unternehmen usw. zu finden. Der Computermarkt war ein reiner Hardwaremarkt, Software war eine Beigabe und bot nur eine rudimentäre Basisausstattung, damit die Hardware überhaupt laufen konnte. Sie wurde zunächst immer mit dem Quellcode ausgeliefert. Die Nutzer mussten die von ihnen benötigte Software entweder selbst schreiben oder die vorhandene an ihre Bedürfnisse anpassen. Die Hardware-Hersteller hatten es mit programmierkompetenten Kunden zu tun, die von ihren Firmen oder Universitäten für das Programmieren bezahlt wurden und nicht für die Programme selbst. Bis Ende der 1960er Jahre dominierten Hardware-Hersteller, vor allem IBM und die „sieben Zwerge“ die Computerindustrie (vgl. Ceruzzi 2003: 182). Erst in den 1970er Jahren entstanden die ersten eigenständigen Softwarefirmen (Grassmuck 2002: 202f.).

Damals hieß Programmieren, den erstellten Programmcode auf Lochkarten oder -streifen zu übertragen und diese dann an ausgewählte Personen zu übergeben, die als einzige die Maschine bedienen durften. Diese legten die Karten in den Computer ein und überreichten schließlich den Programmeschreibern einen Computerausdruck

mit den Ergebnissen. Wenn ein Fehler im Code war, stürzte das System ab und der ganze Prozess musste von vorne begonnen werden (Levy 2001: 26). Anfang der 1960er Jahre fing die Computerwelt an sich zu wandeln. Es kam ein zwischen Mensch und Maschine „interaktiver" Programmierprozess auf, für den die Zwischeninstanz der Bediener nicht mehr notwendig war. Die Programmierer arbeiteten nun selbst am Computer. Wenn etwas schief ging, konnte dies sofort bemerkt und darauf reagiert werden.

3.1.1 Die Hacker am MIT

1959 wurde der erste Computer-Programmierungskurs am *Artificial Intellegence Lab* (AI Lab) des *Massachusetts Institute of Technology* (MIT)[1] angeboten. Zu dieser Zeit sahen selbst am MIT die meisten Menschen Computer nicht als ein eigenständiges Forschungsfeld an, sondern bestenfalls als nützliche Werkzeuge (Levy 2001.: 25). Einige Mitglieder des *Tech Model Railroad Club* (TMRC) am MIT, der sich mit dem Bau komplexer Modelleisenbahnwelten beschäftigte, wurden besonders davon angezogen. Im Unterkomitee für Signal- und Stromversorgung (das *System*) beschäftigten sie sich mit de Funktionieren des Systems der Modelleisenbahnen und begrüßten die zunehmende Komplexität bei jeder Änderung. Ein wichtiger Aspekt war hierbei, die Grenzen des Systems auszutesten . Nun wurden sie begierig darauf zu lernen, wie Computer funktionierten und begeisterten sich für die Tätigkeit des Programmierens (Levy 2001: 24). Während ihrer Arbeit an den Modelleisenbahnen entwickelten sie ihren eigenen Jargon. So war beispielsweise ein *Hack* ein Projekt oder Produkt, das nicht einfach nur ein bestimmtes Ziel erreichte, sondern das mit einem „wilden Vergnügen" bei der Entstehung verbunden war, sich durch Innovation, Stil und technische Virtuosität auszeichnete, aber auch etwas Unorthodoxes an sich hatte.[2] Die meisten am *System* aktiv Beteiligten bezeichneten sich selbst stolz als Hacker (ebd.: 23).[3]

Schnell wurden sie die größte Benutzergruppe des ganzen Computerzentrums (ebd. 26). Es gesellten sich auch Auswärtige zu den TX-0 Hackern, wie sie sich nach ihrem

[1] Nach Eric Raymond war das MIT das führende Forschungsinstitut im Bereich der Künstlichen Intelligenz bis zu Beginn der 1980er Jahre (Raymond 1999b: 20).

[2] Später wurde der Begriff am MIT auch für nicht-technische Aktivitäten benutzt (Levy 2001: 23).

[3] Die Bezeichnung *Hacker* wird mit zweierlei Deutung versehen. Zunächst waren Hacker das, was in diesem Kapitel geschildert wird. In den 1980er Jahren kam eine zweite, negativ besetzte Bedeutung dazu: Menschen, die in fremde Computersysteme einbrechen. Die negative Belegung, wurde einerseits durch die Massenmedien benutzt, andererseits bezeichneten die Betroffenen sich selbst als Hacker (Levy 2001: 432). Es gibt für die Computer-Kriminellen auch die Bezeichnungen *Cracker* oder *Warez dOOdz*, wobei *Cracker* eher in Systeme einbrechen und *Warez dOOdz* eher den Kopierschutz von Programmen umgehen und diese Versionen wieder verteilen (gegen Geld oder auch umsonst) (The Jargon Dictionary 2000a, The Jargon Dictionary 2000b). Es gibt also eine emische und eine etische Bedeutung des Begriffs. Dass sich *Cracker* selbst als *Hacker* bezeichnen, lässt sich wohl darauf zurückführen, dass sie über die etische Bezeichnung ihre Identität zum Teil konstruieren. In dieser Arbeit wird die emische Bedeutung der Bezeichnung *Hacker* verwendet.

bevorzugten Computer nannten. Peter Deutsch, der den Zugang zum AI Lab durch seinen am MIT arbeitenden Vater erhielt, kam schon als Kind dazu. Deutsch wurde trotz seines geringen Alters von den anderen wegen seiner Computerkenntnissen akzeptiert und als gleichberechtigt behandelt (ebd.: 30f.).

Sie arbeiteten auf unorthodoxe Weise, manchmal bis zum Morgengrauen, fuhren dann zu einem chinesischen Restaurant und danach ging es häufig mit dem Programmieren weiter. Ihr Wissen eigneten sie sich durch die Beobachtung der anderen an (was sich zwangsläufig durch die Wartezeit auf die eigene Rechnersession ergab), durch Fragen an ihre Hackerfreunde, wenn diese sich auf einem für sie relevanten Gebiet gut auskannten und – vielleicht am wichtigsten – durch Ausprobieren und Herumspielen. Das Ausprobieren war notwendig, da es nur wenig existierende Software und kaum Handbücher gab. Viele der selbstgeschriebenen Programme waren Entwicklungstools (ebd.: 32). Der Nutzen einiger Programme – beispielsweise eines, dass arabische Zahlen in römische umwandelte – war allerdings nicht so leicht zu erkennen, sie dienten dazu, die Computer zu erforschen. Ein Teil der Gruppe wurde nach einer Weile vom AI Lab als studentische Hilfskräfte und ähnliches angestellt, unter anderem auch deswegen, weil sie sich mit einigen der Rechner am besten auskannten.

Die Gruppe schuf ihre eigenen Wertvorstellungen, die Levy als *Hackerethik* bezeichnet und sich im stillen Einvernehmen herausgebildete (ebd.: 39ff.). Der zentrale Punkt war der freie Zugang zu Computern und Daten. Sie befürworteten offene Systeme, in denen es keine Blockade zwischen ihnen und einer Information oder einem Teil der benötigten Ausrüstung gab. Daher lehnten sie auch offizielle Bürokratien ab, die den freien Zugang stark behinderten – dies verbanden sie mit ihrem kulturellen Feindbild, IBM und deren Krawatte tragenden Ingenieuren. Die Ablehnung von Autoritäten spiegelte sich auch in der Zuweisung von Anerkennung: Titel, Alter und ähnliches waren irrelevant – nur die Qualitiät des Codes zählte. Hierzu gehörte, dass man eleganten Code schrieb. Ein wichtiges Kriterium dafür war, intelligente Lösungen für Probleme zu finden und komplizierte Operationen mit so wenig Anweisungen wie möglich auszudrücken.[4]

Die Hacker am MIT hatten damals nur die Chance sich auszuleben, weil die Leiter des AI Labs ihren Wert erkannten. Diese Professoren ermutigten sie, an ihren Projekten zu arbeiten, und schützen sie auch bei Konflikten mit den *Graduate Students*, die sich durch die nicht offiziell ausgebildeten Hacker gestört fühlten. In den siebziger Jahren wandelte sich die Gruppe: einige gingen zu anderen Computerzentren, neue kamen nach, die die Strukturen, Sitten und Gebräuche, die ihre Vorgängen entwickelt hatten, als gegeben übernahmen.

[4] Dies war sehr wichtig, da zu dieser Zeit Computer nur einen sehr kleinen (Arbeits-)Speicher hatten, so dass schon ein paar Zeilen Code zu viel sein konnten. Je kürzer ein Programm war, desto schneller lief es. Außerdem blieb so auch mehr Platz für andere Programme.

3.1.2 Unix

1969 war ein wichtiges Jahr für die Entwicklung von Betriebsystemen.[5] Die Arbeit an dem Betriebssystem *Multics* (**MULT**iplexed **I**nformation and **C**omputing **S**ervice), einer Kooperation zwischen den damals zur monopolistischen Telefongesellschaft AT&T gehörenden Bell Labs, dem MIT und General Electric, wurde eingestellt. Ken Thompson, der für die Bell Labs an diesem Projekt arbeitete, fühlte sich seines Spielzeugs beraubt, so dass er in nur vier Wochen die erste Version des Betriebssystems *Unix* (**UN**iplexed **I**nformation and **C**omputing **S**ervice)[6] schrieb (Salus 1995: 8ff.). Thompson wollte nach der Erfahrung mit Multics alles etwas kleiner und schlichter halten, trotzdem übernahm er einige Ideen von Multics. Sein Kollege Dennis Ritchie entwickelte eine neue Programmiersprache namens C,[7] um sie unter einem sehr frühen Unix Stadium zu verwenden. Beide, Unix wie auch C, sollten leicht bedienbar, ungezwungen und flexibel sein, die „Keep it simple, stupid" Philosophie, wie es Eric Raymond beschreibt (Raymond 1999a: 23). Programmierer konnten sich die logischen Strukturen leicht merken, ohne dauernd Manuals zu konsultieren zu müssen, wie es vorher meistens der Fall war.

Nachdem Thompson und Ritchie 1973 eine Präsentation über Unix auf dem ACM Symposium on Operation Systems gehalten hatten, begannen auch andere sich für Unix zu interessieren (Salus 1995: 54f.). Da AT&T ein staatlich reguliertes Monopol innehatte, war es ihnen untersagt, sich in anderen wirtschaftlichen Bereichen zu betätigen. Sie konnten Unix also nicht regulär vermarkten, sondern nur eine nominelle Gebühr verlangen (ebd.: 56f.). So entschloss sich AT&T, Unix an Bildungseinrichtungen weiterzugeben,[8] allerdings verweigerten sie jeglichen Support oder Bugfixes. Dies hatte zur Folge, dass den Nutzern nichts anderes übrig blieb, als sich zusammenzuschließen. Sie teilten Ideen, Informationen, selbst geschriebene Programme, Bug- und Hardwarefixes. Die Unix-Kopien für die Installation wurden teilweise persönlich von den Interessierten bei den Bell Labs abgeholt, oder Ken Thompson brachte sie den verschiedenen Institute (ebd.: 65). Nach und nach entstand eine Fülle von Programmen und Unix wurde auf verschiedene Hardware-Plattformen portiert.[9] Ab 1977 gab es eine weitere Unix-Distribution[10], neben der von AT&T: die *Berkeley*

[5] Betriebssysteme bilden die unterste Softwareebene. Sie organisieren den Zugriff auf Festplatten, Speicher, etc. und stellen Anwendungsprogrammen diese Funktionen bereit.

[6] Schon bald änderte sich die Schreibweise in Unix. Wer für diese Änderung verantwortlich ist, ist heute nicht mehr klar (Salus 1995: 9).

[7] Ritchie erweiterte eine Sprache namens „B", bis er ihr auch einen neuen Namen gab: C (Salus 1995: 48f.).

[8] Offenbar ohne eine Gebühr zu erheben (vgl. Salus 1995: 65).

[9] Es ist nicht möglich, ein Betriebssystem, dass für einen bestimmten Rechner geschrieben ist, einfach auf einen anderen zu installieren. Es muss portiert werden, d.h. dass es entweder völlig neu für diese Architektur geschrieben werden muss, oder es müssen einige große Anpassungen vorgenommen werden.

[10] Eine Distribution ist eine Zusammenstellung von einzelnen Softwareprogrammen, die im bestens Fall auch noch so verbunden sind, dass sie sich relativ einfach zusammen installieren lassen.

Software Distribution (BSD). Sie wurde von Bill Joy, einem Doktoranten an der Universität von Kalifornien in Berkeley und späteren Mitbegründer der Computerfirma *SUN* (Stanford University Network) *Microsystems* zusammengestellt und umfasste Programme für Unix, die in Berkeley entwickelt wurden (McKusick 1999: 33). Es gab also zwei Unix Distributionen, die von AT&T und die von BSD.

Die Zusammenarbeit und Vernetzung der Unix-User über einzelne Institute hinweg erfolgte auf mehreren Ebenen. Ab 1975 gab es die *Unix NEWS,* einen Newsletter der Unix-User, der sich zwei Jahre später umbenennen musste, da die Rechtsanwälte von AT&T den Namen als einen Verstoß gegen das Trademark „Unix" ansahen.[11] Daher nannte sich eine 1978 gegründete Unix User Gruppe *USENIX* (Salus 1995: 68f.).[12] Neben USENIX gründeten sich verschiedene regionale Benutzergruppen in Europa und Asien. Die technische Vernetzung von Rechner zu Rechner bzw. Institut zur Institut erfolgte 1978 mit dem *Unix-to-Unix Copy Protocol* (UUCP), das es ermöglichte, zwei beliebige unixbetriebene Maschinen miteinander über normale Telefonleitungen zu verbinden. Die Institute mit Unix-betriebenen Rechnern entwickelten sich zu einer eigenen *Netzwerknation* (Raymond 1999a: 24). Erst 1983 wurde das *Transmission Control Protocol/Internet Protocol* (TCP/IP) Protokoll an Unix angepasst und in die Berkeleyversion integriert, die zu der Zeit von DARPA benutzt wurde. So wurde auch für die Unix Benutzer die Nutzung des ARPAnets bzw. des Internets möglich.[13]

Da Unix die gleiche Gestalt und die gleichen Fähigkeiten bei vielen verschiedenen Arten von Maschinen hatte, konnte es als gemeinsame Softwareumgebung dienen. Dies bedeutete, dass durch den Wegfall einer Maschine der Nutzer nicht mehr gezwungen war ein völlig neues Softwaredesign zu kaufen, bzw. zu entwickeln (Raymond 1999a: 22f.). Aus diesem Grund entschied sich auch die *Defence Advanced Research Projects Agency* (DARPA)[14] im Jahre 1979 für die Berkeley Variante von Unix, die 3BSD Release.

[11] Hieran lässt sich erkennen, dass AT&T sich langsam etwas mehr für Unix zu interessieren begann.

[12] USENIX hält unter anderem Treffen ab und es gibt sie noch heute.

[13] Das ARPAnet wurde 1969 entwickelt. Es war das erste transkontinentale Hochgeschwindigkeits-Computernetzwerk und der Beginn des Internets. Es entstand als Reaktion auf den „Sputnik-Schock" der Amerikaner. 1958 wurde die *Advanced Research Project Agency* (ARPA) gegründet, um den technologischen Vorsprung der USA im Militärbereich zu sichern. Ein Projekt war eben das ARPAnet, dessen Ziel es unter anderem war, ein Computernetz zu schaffen, dass dezentral und nicht hierarchisch aufgebaut war, so dass im Falle eines Angriffs die Chancen eines Totalausfalls der Kommunikation verringert werden konnte. Es wurde vom Verteidigungsministerium in Auftrag gegeben. Die technische Entwicklung und Implementierung erfolgte durch wissenschaftliche Einrichtungen. Es war später mit Hunderten von Universitäten, Rüstungsunternehmen und Laboratorien vernetzt. 1976 wurde das TCP/IP entwickelt, es ist bis heute das Basisprotokoll des Internets auf dem die einzelnen Dienste (zu Beginn: Mail, FTP, Usenet, Telnet) laufen. 1983 spaltete sich das MILnet, das Militärnetz, vom ARPAnet ab (vgl. Raymond 1999: 20, Castells 2001: 50, Salus 1995: 163f., Becker 2002: 24f.).

[14] ARPA und DARPA ist die gleiche Organisation, die abwechselnd einen der beiden Namen trug (vgl. DARPA 2003). Hier wird immer die zum jeweiligen Zeitpunkt aktuelle Bezeichnung verwendet.

3.1.3 Computers to the people

Die vorangegangenen Beschreibungen befassten sich mit Menschen, die bereits einen Zugang zu Computer hatten. Um diesen Zugang zu erreichen, musste man aber an eine Universität, an eine Forschungseinrichtung oder an ein entsprechendes Unternehmen angebunden sein. Diese Beschränkungen trieb ab Anfang der 1970er Jahre Hardwarebastler dazu, ihren eigenen, kleinen Computer zu bauen. Dies wurde unter anderem durch fallende Preise im Bereich der Hardware ermöglicht, aber auch durch die Leistungssteigerung von kleinen Prozessoren (Freiberger/Swaine 2000: 41f.).[15] Es gab eine Konzentration der Hardwarehacker in der Bay Area um San Fransisco. Dies kam einerseits durch das Silicon Valley, wo viele bei Unternehmen, wie Xerox Parc oder Hewlett-Packard (vgl. Kaplan 2000; Hiltzik 2000), arbeiteten und andererseits durch Menschen, die durch die alternative und politisierte Lebensweise in der Region geprägt waren und die daher den Nutzen von Computer für die Allgemeinheit bereitstellen wollten (vgl. z. B. Young 1989: 63ff.).

Der erste so genannte Mikrocomputer, der Altair 8008, wurde 1975 von *Micro Instrumentation Telemetry Systems* (MITS), einer Firma in Albuquerque, New Mexico, entwickelt. MITS vertrieb den Altair als Bausatz. Er hatte keinen Monitor, statt dessen musste man die Prozesse von Leuchtdioden ablesen. Es gab zunächst auch keine Software für ihn. Diese und Hardwareerweiterungen wurden nach und nach von den Nutzern selbst entwickelt und an die anderen weiterverteilt – teilweise in kommerzieller Art, indem sie kleine Start-Ups gründeten. Zunächst war es nicht möglich, etwas „Vernünftiges“ mit dem Altair anzufangen, aber man konnte ihn erforschen (Freiberger/Swaine 2000: 33ff.). Der Altair war nicht der einzige Versuch, einen Mikrocomputer selbst zu entwickeln, ohne dass die Intention von der Forschungsabteilung einer großen Firma ausging. Die bestehenden großen Computerfirmen sahen zunächst keinen Markt für Kleincomputer für den Heimgebrauch. Damit überließen sie kleinen Start Ups das Feld, das bekannteste Beispiel ist Apple (vgl. Young 1989; Freiberger/Swaine 2000: 149). Binnen weniger Jahre entwickelten sich diese „Garagenfirmen“ zu einer Multi-Milliarden-Dollar-Industrie. Neben den Hobbyisten entstand eine zweite Gruppe von Käufern, Kunden aus der Geschäftswelt. Diese erwarben Computer als Hilfsmittel, nicht als intellektuelles Spielzeug. Seit der Vorstellung des IBM *Personal Computers* (PC) [16] 1980 entwickelten sich daraus zwei getrennte Märkte. Auf der einen Seite standen Firmen wie Commodore, Atari, Amstrad und Sinclair mit kleinen, relativ preiswerten Rechnern für den Heimbereich. Auf der anderen Seite stand IBM mit seiner für Büroanwendung konzipierten Architektur. Da IBM diese öffentlich machte, entwickelte sich eine Gruppe von Herstellern

[15] Diese waren in keiner Weise der Leistung von Prozessoren der Großrechner vergleichbar.

[16] Eigentlich bezeichnet „PC“ alle Computer für Endbenutzer und stellt den Gegenbegriff zu „Großrechner“ dar. Schnell hat es sich jedoch als Synonym für Systeme auf Basis der IBM Mikrocomputerarchitektur entwickelt – und steht damit eher im Gegensatz zu „Apple“.

(z. B. Compaq, Wang, Olivetti) so genannter Klone, die untereinander kompatibel waren. Diese Nachbauten machten die IBM-Architektur zum Industriestandard (Allner/Allner 2003: 62ff.; Grassmuck 2002: 204).

Zwar gab es auch in der Phase des Heimcomputers als Hobbygerät schon kommerzielle Software (beispielsweise die Programmiersprache BASIC von Microsoft), aber das gros der Software schrieben sich die Computerbesitzer selbst und tauschten es untereinander aus. Als erstes Signal einer Kommerzialisierung der Software kann man Bill Gates' „Open Letter to Hobbyists" aus dem Jahre 1976 ansehen. Darin beklagt er einerseits die Verbreitung von Raubkopien, andererseits argumentiert er, dass gute Programme nur dann geschrieben werden, wenn ihre Entwickler damit auch Geld verdienen können (Freiberger/Swaine 2000: 195; Gates 1976). Zunächst wurde professionelle Software noch von den Computerherstellern als Beigabe mitgeliefert (ähnlich wie zu Beginn im Bereich der Großrechner). Doch bald entstanden die ersten reinen Softwarefirmen, wie Visicorp, die eine Tabellenkalkulation für den Apple II in sehr hoher Stückzahl verkauften. Im Markt der IBM kompatiblen PCs schritt diese Entwicklung rasch voran. Softwarehäuser wurden nun zunehmend das Zentrum der Innovation (Ceruzzi 2003: 314). Microsoft machte dabei einen besonders guten Deal: ihr Vertrag mit IBM über die Lieferung von MS-DOS als das Betriebssystem für den PC sah nicht nur eine Lizenzgebühr für jeden verkauften PC vor, sondern gestattete es ihnen auch, ihre Rechte zu behalten und so ihr Produkt an andere Hersteller weiter zu lizensieren. Parallel zum offenen Hardwarestandard entstand so das proprietäre Softwaremonopol, das Microsoft schnell zur größten Softwarefirma machte (Freiberger/Swaine 2000: 328ff.).

Die Entwicklung des PC ist letztlich eine Radikalisierung des Hackerideals des freien Zugangs. Nun bezog sich diese Forderung nicht mehr auf die eigene Person und bestehende Ressourcen, sondern der ganzen Menschheit sollte die Möglichkeit gegeben werden, sich mit Computern zu beschäftigen. Die Art des Kontakts unter den PC-Benutzern gestaltete sich allerdings anders, als bei den Großrechnerbenutzern. Die PCs standen zu Hause im Wohn- oder Kinderzimmer und waren damit rund um die Uhr für einen alleine verfügbar. Die Beschäftigung mit den Computern verlagerte sich also von öffentlichen Orten in den privaten Raum, was zur Folge hatte, dass die Benutzer sich von einander isolierten. Die direkten Kontakte zu Gleichgesinnten geschahen nun eher im kleinen Rahmen. Eine weiter reichende, direkte Vernetzung lief über die wenigen Clubs, die sich sehr bald gründeten. Einer der ersten war der *Homebrew Computer Club*, dieser beeinflusste auch die Entstehung der PCs stark. In Deutschland – allerdings eher in den 1980er Jahren – ist der *Chaos Computer Club* (CCC) zu nennen. Aber anders als bei Forschungseinrichtungen, Unternehmen usw. handelte es sich bei diesen Clubs um Gruppen, deren alleiniger Zweck die Vereinigung Menschen mit gleichen Interessen war. Ein weiteres wichtiges, allerdings indirektes, Vernetzungsmedium waren Zeitschriften. Es gab allgemeine Zeitschrif-

ten, andere, die sich nur mit einem bestimmten Heimcomputer befassten und solche, die für Nutzer, die ihren PC als Werkzeug ansahen. Die Magazine waren eine sehr wichtige Informationsquelle, sie veröffentlichten auch seitenweise Quellcodes von Programmen, die man abschreiben konnte (Freiberger/Swaine 2000: 109ff., 211ff.). Der direkte Kontakt über weitere Distanzen hinweg war im großen Stil erst mit der Öffnung des Internets für die Allgemeinheit möglich, welche durch die Entstehung von kommerziellen Providern Anfang der 1990er Jahre begann. Mit dem World Wide Web und Email standen ab diesem Zeitpunkt zwei einfach zu bedienende Werkzeuge auch für die breite Masse zur Verfügung (vgl. Ceruzzi 2003: 345ff.).[17]

3.2 Die Befreiung der Software

Da Software sich zu einer eigenständigen Ware entwickelte, folgte auch die Exklusivierung des Quellcodes, d.h. er wurde gar nicht mehr mitgeliefert oder nur gegen eine sehr hohe Gebühr und wurde nun als Geschäftsgeheimnis behandelt. Ein Faktum, das der Hackerethik völlig widersprach und auf das es Gegenreaktionen gab. Die wohl bekannteste ging von Richard Stallman aus, einen damaligen Mitarbeiter am AI Lab. Ein anderes Beispiel ist BSD, die Unix Distribution der Berkeley Universität. Dies sind die beiden bekanntesten und prägendsten Beispiele von bewußt als frei konzipierter Software, auch wenn es noch weitere gibt, deren Beschreibung den Rahmen dieser Arbeit sprengen würde.

3.2.1 Freie Software

Anfang der achtziger Jahre endete die Zeit der Hacker am MIT. Einige der besten Programmierer des Labors wechselten, bedingt durch die beginnende Kommerzialisierung der Software, zu Start-Up-Unternehmen und in hoch bezahlte Jobs. Die vom AI Lab benutze Hardware veraltete und wurde vom Hersteller eingestellt. ITS, das Betriebssystem der Rechner im AI Lab, war an die alte Hardware gebunden und es gab nicht mehr genug qualifizierte Programmierer, die es auf die neue Hardware hätten portieren können. Außerdem gab es noch eine weitere Änderung: die Unternehmen gaben ihrer Software nicht mehr den Quellcode bei. So war es nicht mehr möglich sich selbst zu helfen, wenn ihre Software fehlerhaft war.[18]

[17] Es gab zwar schon seit Mitte der 1980er die Möglichkeit als Privatperson am Internet teilzunehmen, dies war erstens technisch etwas anspruchsvoller und recht teuer. So nutzten es nur wenige PC-Benutzer. *Internet Service Provider* (ISP) bieten Nutzern an, sich gegen eine Gebühr mithilfe eines Modems über eine normale Telefonleitung einzuwählen. die Server des ISP sind mit dem Internet verbunden, finden dort die gewünschten Daten und senden sie über die Telefonverbindung zum Endnutzer zurück.

[18] Software ist eigentlich immer fehlerhaft. Ein der ganz wenigen Ausnahmen dürfte *TEX* von Donald Knuth sein (vgl. Moody 2001: 221).

Eine Person, die am AI Lab übrig blieb, war Richard Stallman.[19] Er musste nicht nur die Auflösung seines sozialen Umfeldes erleben, sondern empfand die Geheimhaltung des Quellcodes als antisozial, unethisch und einfach falsch, da sie es unmöglich machte, seinen Nachbarn zu helfen und Kooperation verhinderte (Stallman 1999: 54). Stallman stand also vor der Frage, was zu tun sei. 1983 beschloss er ein *Freies* Betriebssystem zu schreiben und so wieder eine Gruppe von kooperierenden Hackern aufzubauen, der jeder beitreten konnte (ebd.: 55). *Freie* Software musste für Stallman folgende Kriterien erfüllen (ebd.: 56):

1. Man muss die Freiheit haben, das Programm zu benutzen und zwar zu jedem Zweck.

2. Man muss die Freiheit haben, das Programm zu verändern, so dass man es seinen Bedürfnissen anpassen kann. Hierfür braucht man natürlich den Quellcode.

3. Man muss die Freiheit haben, Kopien des Programms zu verteilen, entweder gratis oder gegen eine Gebühr.

4. Man muss die Freiheit haben, Kopien des veränderten Programms zu verteilen, so dass die Allgemeinheit (bei Stallman heißt es Community) von den Verbesserungen profitieren kann.

Dieses freie Betriebssystem sollte kompatibel mit Unix sein, denn so wäre es portierbar und anderen fiele der Wechsel leicht. Außerdem könnte ein portierbares System in Zukunft verhindern, dass die Software durch eine Änderung der Hardware obsolet wird. Als Namen wählte er *GNU*, was für *GNU's Not Uni*x steht. Im Januar 1984 kündigte Stallman seine Anstellung am MIT, damit das MIT keine Rechte an GNU für sich beanspruchen konnte. Trotz seiner Kündigung wurde Stallman erlaubt, sein Büro und die Infrastruktur dort weiter zu benutzen (eine Zeit lang wohnte er sogar in seinem Büro) (ebd. : 57). Zum Einstieg in GNU schrieb Stallman 1985 Emacs völlig neu (Williams 2002: 122). Da es schon vorher ein bekanntes Programm war, fand es auch jetzt wieder recht leicht neue Nutzer und konnte so die Aufmerksamkeit auf das GNU Projekt ziehen. Zunächst machte es Stallman per FTP zugänglich, später, da

[19]Stallman kam 1971, noch als Student der Universität Harvard, ans AI Lab. Seine Aufgabe bestand darin, das Computersystem leistungsfähiger zu machen. So schrieb er 1975 ein Editor namens *Editing MACroS* (Emacs). Dieses Programm verbreitete sich schnell über die Grenzen des MITs hinweg. Mit den Kopien verschickte Stallman die "informellen Regeln", die besagten, dass jeder der Veränderungen vornahm, diese an Stallman zurückschicken müsse. Laut Moody wurde die Regel nach ihrer Durchsetzung zur Grundlage der gesamten Bewegung der Freien Software und einer der entscheidenden Faktoren für ihren Erfolg. Als Motivation für Stallmans Handeln sieht Moody, dass die Kopien auch an Menschen außerhalb des MITs verschickt wurden, also an Leute, für die es nicht selbstverständlich sein musste, dass sie ihre Ergebnisse weitergeben würden (Moody 2001: 29f.). Emacs ist auch noch heute einer *der* Editoren. Ein Editor ist ein Programm zu Eingabe von Texten, das besonders auf die Bedürfnisse der Quellcodeerstellung und nicht so sehr auf Textverarbeitung ausgerichtet ist.

nicht alle Interessenten einen Internetzugang hatten, kopierte er es für eine Gebühr von US$ 150 auf Diskette und schickte es den Betreffenden zu. So verdiente er seinen Lebensunterhalt. Er veröffentlichte es unter der *GNU Emacs License* (ebd.: 124). Diese schreibt nicht nur die Zugänglichkeit des Quellcodes vor, sondern auch, dass jede darauf basierende Software wieder unter der gleichen Lizenz veröffentlicht werden muss. Dieser Vererbungseffekt wird von Stallman als *Copyleft* bezeichnet. Den Begriff *Copyleft* übernahm Stallman von einem Aufkleber auf dem stand: „Copyleft (L), All Rights Reversed." (ebd.: 128).[20] So kam es, dass Richard Stallman mit der *GNU Emacs License Freie Software* explizit ins Leben rief. Später, 1989, wurde die Erweiterung der *GNU Emacs License*, die Version 1.0 *GNU General Public License* (GPL) veröffentlicht.

1985 gründeten Stallman und einige Mitstreiter die *Free Software Foundation* (FSF)[21], deren Präsident Stallman heute noch ist. Die FSF übernahm die geschäftliche Seite des GNU Projekts, einschließlich der rechtlichen Vertretung, wenn es zu Lizenzverstößen seitens anderer kam. Außerdem betätigt sich die FSF auch im Bereich der Lobbyarbeit, um die Verbreitung Freier Software zu fördern (Stallman 1999: 60f.).

3.2.2 BSD

Die erste und zweite Release von BSD, der Unix Distribution der Universität Berkeley, enthielt nur Zusatzprogramme für Unix im allgemeinen. Ab 3BSD aus dem Jahre 1979 enthielt sie auch Originalcode von AT&T. Für die Lizenz einigten sich die AT&T Anwälte und die Universität Berkeley darauf, dass die Nutzer zusätzlich auch immer eine AT&T Lizenz erwerben müssten. Außerdem konnte die Software von der Berkeley Universität mit Quell- und Binärcode[22], mit oder ohne Veränderungen weitergegeben werden, allerdings mussten der Copyright-Vermerk der Universität Berkeley oder einzelner Beitragender und der Lizenztext mit weitergegeben werden (Grassmuck 2002: 279). Neben BSD gab es einige (Hardware)Hersteller, die ihre eigenen, kommerziellen und teilweise proprietären Unix-Versionen veröffentlichten, beispielsweise AIX von IBM, SCO-Unix von *Santa Cruz Operation* (SCO) und SUN/OS und Solaris von SUN. Allerdings waren diese kommerziellen Versionen untereinander nicht kompatibel, d. h. man kann nicht ein Programm für AIX auf

[20] Gegner von Freier Software beispielsweise Microsoft (vgl. Babcock 2001, Shankland 2001) denunzieren den Copyleft-Effekt auch als *viral* (von Virus abstammend, also eine Verbindung mit einem krankheits-, wenn nicht todbringendem Phänomen), eine Bezeichnung, die in einigen wissenschaftlichen Studien zum Thema (z.B. Osterloh/Rota/Kuster 2004: 126ff.) unkritisch übernommen wird.

[21] http://www.fsf.org

[22] Während Quellcode ein Programm in für Menschen lesbarer Form – nämlich als Anweisungen in einer Programmiersprache – meint, bezeichnet Binärcode die daraus – meist durch einen Compiler – generierte, für den Computer ausführbare Form. In Binärcode lassen sich also nachträglich keine Veränderungen mehr vornehmen, da er nicht mehr verständlich und eindeutig ist. Er lässt sich nur als fertiges Produkt verwenden und kopieren.

Solaris laufen lassen.

1984 wurde AT&T aufgespalten. Nun konnte das Unternehmen Unix kommerziell vermarkten. Zu diesem Zweck wurde eine Tochterfirma namens *Unix System Laboratories* (USL) gegründet, welche auch die Unix Lizenz veränderte (ebd.: 215). Der Preis für die AT&T Lizenz stieg nun kontinuierlich. Als Folge traten die Nutzer an BSD heran, und baten sie, eine Möglichkeit zu finden, die AT&T Lizenz überflüssig zu machen. Daraufhin wurde bei der *Networking Release 1* im Jahre 1989 der AT&T Code gänzlich herausgenommen. *Networking 1*[23] war die erste frei weiterverteilbare Release von Berkeley. Die einzige Vorschrift war, dass bei der Weitergabe der Copyrightvermerk intakt bleiben musste und bei veränderten Versionen in der Dokumentation der Hinweis vorhanden war, dass das Produkt auf Code der Universität von Kalifornien und deren Beitragenden basierte (McKusick 1999: 40f.). Das bedeutet, dass die BSD Lizenz im Gegensatz zur GPL keinen Copyleft- oder Vererbungseffekt hatte.

Schließlich wurde begonnen, den noch fehlenden AT&T Code neu zu schreiben, d.h. es wurde ein Code geschrieben, der die gleiche Funktionalität, wie der alte hatte. Dabei orientierten sich die Programmierer an den veröffentlichten Beschreibungen. 1992 war dies vollbracht. BSD war nun ein voll funktionstüchtiges und freies Betriebssystem (McKusick 1999: 42f.). Kurz darauf wurde die 386/BSD Release für die 386 PC-Architektur veröffentlicht und auf einen anonymen FTP-Server gestellt, so dass jeder sie sich umsonst herunterladen konnte. Da der Initiator von 386/BSD mit der Integration der Fehlerbehebungen nicht hinterher kam, gründete eine Gruppe von 386/BSD Nutzer das freie Projekt NetBSD,[24] deren erste Release, *NetBSD 0.8*, 1993 herauskam (NetBSD Foundation 2004). NetBSD legte seinen Schwerpunkt auf eine möglichst weitreichende Unterstützung der verschiedensten Hardware-Plattformen. Ein anderes Projekt, FreeBSD[25], gründete sich kurz nach NetBSD. FreeBSD interessierte sich in erster Linie für eine bessere PC Unterstützung (McKusick 1999: 42). 1995 spaltete sich dann noch OpenBSD[26] aufgrund persönlicher Konflikte von NetBSD ab. Der thematische Schwerpunkt von OpenBSD liegt auf Sicherheit. Außer NetBSD, FreeBSD und OpenBSD gibt es noch weitere kleinere freie BSD Projekte, beispielsweise MirBSD[27].

Neben diesen freien Projekten gab es auch kommerzielle Weiterentwicklungen der BSD Software durch die Firma *Berkeley Software Design, Inc.* (BSDI). BSDI verkaufte ihre Version im Jahre 1992 für US$ 995, während USL zu der Zeit für das AT&T Unix samt Quellcode US$ 100 000 verlangte. Daraufhin verklagte USL zu-

[23] *Networking 1* war keine vollständige Veröffentlichung des gesamten BSD Unix. Es handelte sich hier um die Software, die das TCP/IP und die sonstige Internetunterstützung enthielt.
[24] http://www.netbsd.org.
[25] http://www.freebsd.org
[26] http://openbsd.org/
[27] http://mirbsd.de/

nächst BSDI, dass sie ihre Software nicht Unix nennen durften und später, dass in der BSDI Software USL (AT&T) Code enthalten sei. Da die Software von BSDI auf dem BSD Code aufbaut, wurde nun auch die Berkeley Universität von USL verklagt. Daraufhin reichte Berkeley eine Gegenklage gegen USL ein, da in deren Code auch BSD Code eingearbeitet sei. Der Rechtsstreit endete erst, als USL von Novell gekauft und schließlich 1994 mit der Berkeley Universität ein Vergleich erzielt wurde (ebd.: 44f.).

3.3 http://www.linux.etc

1991 entwickelte der am CERN, dem europäischen Kernforschungszentrum in Genf, arbeitende Tim Berners-Lee das *World Wide Web* (WWW).[28] Nachdem das Verbot der Nutzung des Internets für Werbung aufgehoben wurde, entstand in den 1990er Jahre die Industrie der Internetprovider, die Zugänge zum Internet für ein paar Dollar im Monat an jedermann verkauften und damit erstmals der breiten Masse die Möglichkeit eines Internetzugangs bot. In der Folge wuchs das Internet rapide (Raymond 1999a: 28). Zudem wurden die PCs mit einem Intel 386er Chip (ein im Vergleich zu seinen Vorgängern ein erheblich leistungsstärkerer Prozessor) deutlich günstiger (Torvalds/Diamond 2001: 58).

Im Jahre 1991 fing der finnische Student Linus Torvalds an, den Unix-kompatiblen Kernel[29] Linux zu schreiben. Dies war kein geplanter Vorgang, sondern diente ihm dazu, den 386er besser kennenzulernen. Er programmierte einige kleine nützliche Werkzeuge für seinen eigenen Gebrauch, als er feststellte, dass er gerade dabei war ein Betriebssystem bzw. den Kernel dazu zu schaffen. Seine ersten Mitentwickler fand er, als er über sein kleines, privates Projekt in der Minix-Newsgroup[30] berichtete (Torvalds 1991). Einige der Newsgroupteilnehmer und Minixbenutzer waren frustriert, da Andrew Tannenbaum, der Autor von Minix, nicht allen aufkommenden Verbesserungsvorschlägen folgte (Torvalds/Diamond 2001: 59 ff.). Neben dieser Frustration, gab es noch zwei weitere Umstände, die für die frühe Entwicklung von Linux förderlich waren: HURD, der lange geplante und erwartete Kernel des GNU-Projekts ließ auf sich warten, und der bereits erwähnte Rechtsstreit um BSD schreckte viele

[28] Das www ist neben Mail, FTP, Usenet, u.a. *ein* Internetdienst, der auf dem TCP/IP Protokoll basiert. Zum WWW gehören *Universal Resource Indentifier* (erst URI, nun URL), *Hypertext Transfer Protocol* (HTTP) und *Hypertext Markup Language* (HTML) (Berners-Lee 1999: 52f.) Fälschlicherweise wird das www manchmal mit dem Internet gleichgesetzt.

[29] Ein Kernel ist der Kern eines Betriebssystems, der die grundlegenden Hardwarefunktionen steuert. Er ermöglicht allen anderen Softwareebenen erst den Zugang zur Hardware.

[30] Minix wurde von Andrew S. Tanenbaum als ein Lehrbetriebssystem geschrieben. Als AT&T begann, Unix kommerziell zu vermarkten, verboten sie, dass der Code zum Unterrichten in Kursen gezeigt werden darf, um so den Status eines Geschäftsgeheimnisses zu wahren. Daraufhin stellten viele Universitäten das Unterrichten von Unix ein oder behandelten es rein auf einer theoretischen Ebene. Andrew Tanenbaum, verfolgte einen anderen Weg, er schrieb ein neues, mit Unix kompatibles Betriebssystem: Minix (Salus 1995: 151f.).

potentielle BSD-Mitentwickler ab. Außerdem brauchte BSD eine leistungsfähigere Hardware als Linux, so dass einigen keine Wahl blieb und sie sich Linux zuwandten (Moody 2001: 94). Ein wichtiger Faktor für das schnelle Anwachsen des Projekt war auch die Art der Projektführung durch Linus Torvalds: Er reagierte schnell auf Fehlermeldungen, Vorschläge für deren Behebung und Ideen für die weitere Entwicklung (ebd.: 95ff.). So wurden eine enorme Anzahl von freiwilligen Mitarbeitern auf der ganzen Welt eingebunden.

Zunächst hatte Torvalds eine eigene Lizenz für Linux geschrieben, die die kommerzielle Verbreitung von Linux untersagte. Erst Version 0.12 wurde im Januar 1992 unter der GPL veröffentlicht (Torvalds/Diamond 2001: 105f., Short/Short 2003). Hintergrund des Lizenzwechsels waren Anfragen, ob es möglich sei für das Kopieren von Linux auf Disketten eine geringe Gebühr zu verlangen. Der Wechsel zur GPL ermöglichte auch die aufkommende Kommerzialisierung von Linux ab 1992. Es entstanden eine Reihe von Firmen, die Linux-Distributionen zusammenstellten, beispielsweise Turbolinux 1992, Red Hat 1993, Suse 1993, Caldera 1994, Mandrake 1998 (Young/Goldman Rohm 2000: 33ff., Turbolinux o. J., Mandrakesoft o. J., Suse 2003, Moody: 134ff.) . Neben den kommerziellen Distributionen entstanden auch freie Projekte wie *Softlanding Linux System* (SLS) 1992, Debian[31] 1993 oder auch Gentoo[32] 2002 (Moody 2001: 122ff.).[33]

Die Distributionen brachten Linux und Freie Software auch Personen näher, die sich nur bis zu einem gewissen Grad mit der Technik auskannten. Das wurde erst durch die Entstehung von Projekten möglich, die sich nicht nur an den Programmiererbedürfnissen orientieren, sondern auch in sehr starkem Maße an denen der reinen Computernutzer. Zwei wichtige Beispiele sind hier die beiden größten Desktopsysteme,[34] *K Desktop Environment* (KDE)[35] und Gnome[36]. Daneben gibt es Versionen der gebräuchlichen Büro- und Internet-Software, die ähnlich einfach zu bedienen sind wie die kommerziell-proprietären Systeme.

Freie Software erfuhr eine zunehmende Verbreitung, in deren Zusammenhang sich

[31] http://www.debian.org

[32] http://www.gentoo.org

[33] Diese Distributionen enthalten nicht nur den Linuxkernel, sondern auch viele weitere Programme, zum Beispiel von den GNU- und X-Projekten. Die Arbeit um eine Linux-Distribution zusammenzustellen bedeutet nicht nur, dass der Quellcode kompiliert wird, sondern die Programme müssen aneinander angepasst werden, d. h., es muss u. a. eine gemeinsame Verzeichnisstruktur geschaffen werden. Außerdem schreiben viele Distributoren Installationsprogramme, z. B. Yast von Suse. Nicht alle Programme in den Distributionen sind Freie Software, da manche nicht in dieser Form existieren, und es Nutzer gibt, die diese Programme gerne unter Linux verwenden möchten und die Distributoren das berücksichtigen. Wie dies genau gehandhabt wird, hängt von den einzelnen Distributoren und ihrer jeweiligen Politik ab, Debian nimmt zum Beispiel nur Freie Software in die Distribution auf und stellt nicht-freie Programme getrennt dazu zur Verfügung.

[34] Desktops sind die graphischen Benutzeroberflächen, mit denen der Computer durch das Anklicken von Menüs und Icons mit der Maus gesteuert werden kann. Dem gegenüber müssen bei Kommandozeileninterpretern (Shell) Befehle eingegeben werden.

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

[36] http://www.gnome.org

eine Vielzahl von Geschäftsmodellen entwickelte. Außerdem organisierten sich Linuxbenutzer in zahlreichen lokalen *Linux User Groups* (LUGs), nach dem Vorbild der *Unix User Groups*. Die weitere Vernetzung der Linuxanhänger geschieht durch Magazine zu diesem Thema und regelmäßigen Treffen von Entwicklern, Linux-Unternehmern und Benutzern auf Konferenzen bzw. Messen.

3.4 Zusammenfassung

Freie Software als bewusste Gegenposition zur proprietären entstand erst in den 1980er Jahren. Sie war eine Reaktion auf die Kommerzialisierung, mit der eine Geheimhaltung des Quellcodes und ein rapider Preisanstieg einhergingen. Hintergründe dieser Ablehnung lassen sich in der *Hackerethik*, wie sie Steven Levy bezeichnet, finden: Die Konzentration auf die zentrale Sache – den Code; die Zuschreibung von Reputation für die Produktion qualitativ hochwertigen Codes aufgrund eines geteilten Verständnisses von Eleganz; die Bewertung von Ergebnissen ohne Ansehen der formalen Qualifikation.

Grundvoraussetzung für diese Ethik ist aber der Zugang zu Hardware und Daten. Genau dieser wurde nun verwehrt. Die Hacker wehrten sich, indem sie eigene Software schrieben. Diese konnte – wie BSD – lediglich selbst frei sein, ohne anderen Beschränkungen aufzuerlegen. Oder – wie Richard Stallman es forderte – das grundlegende Problem des geheimen Quellcodes nachhaltig zu lösen, dass durch die Unterbindung der freien Kooperation den Geist der Gemeinschaftlichkeit zerstört und schlechteren Code nach sich zieht. Diesem Zwecke dient das Copyleft. Es gibt also zwei unterschiedliche Wege innerhalb der Freien Software (vgl. 4.1.2), eine die sich auf Stallman bezieht und eine, die von BSD abgeleitet ist. Aber beiden haben ihre Wurzeln in der Tradition der Hackerethik.

Die Hardwarebastler, die den PC ermöglicht haben, waren ebenso von dem Gedanken des freien Zugangs geleitet. Sie wollten es allen ermöglichen, sich mit einem Computer zu beschäftigen. Die von den frühen Unix-Usern eingeführten Formen der Vernetzung ermöglichten dann die Zusammenarbeit über weite Entfernungen. Auch hier haben sich die Entwickler – wie etwa Tim Berners-Lee, der Entwickler des World Wide Web – sich für offene Standards einsetzt, damit nicht *ein* Unternehmen die Spielregeln diktieren und so die Nutzung und den Nutzen für die Allgemeinheit einschränken konnte.

Der Geist des Ausprobierens und die permanente Verbesserung der Programme durch reges Feedback sind ebenfalls Entwicklungen aus der Frühzeit des interaktiven Programmierens. Kommunikationsformen wie Newsletter und Nutzergruppen stammen auch aus dieser Zeit und sind die Umsetzung des Gedankens des Austauschens, der Offenheit, und des Teilen. Dies schafft auch den Rahmen für einen fairen

Wettbewerb um die beste Lösung und die entsprechende Reputationszuweisung.

Erst durch diese vorausgegangenen normativen und technischen Entwicklungen entstand eine kritische Masse von Personen, die für die Entwicklung eines Softwaresystems wie Linux nötig ist. Aber die frühen Hacker haben nicht nur bis heute gültige Prinzipien und Arbeitsweisen eingeführt. Sie stellen für ihre Nachfolger auch ein Identifikationsangebot dar: Ihre Ideen und ihre Ethik werden durch in kollektive Gedächtnis eingegangene Geschichten und Anekdoten weitertansportiert, die wiederum der Herausbildung und Festigung einer Hackerkultur dienen.

Kapitel 4

Abgrenzung

Freie Software grenzt sich auf der rechtliche Ebene klar von proprietärer Software ab, nämlich durch die spezielle Ausgestaltung der Lizenzen. Diese bewegen sich alle innerhalb des Urheberrechts und nutzen so die rechtsstaatliche Absicherung.[1]

Neben dieser objektiven, rechtlichen Abgrenzung existiert eine subjektive, kulturelle. Diese unterscheidet Insider gegenüber dem Rest der Welt.

Schließlich soll in diesem Kapitel noch auf die inneren Abgrenzungen bzw. die Fraktionierung innerhalb der Entwickler Freier Software eingegangen werden.

4.1 Die rechtliche Ebene

4.1.1 Eine kurze theoretische Einführung in das Thema Eigentum

Umgangssprachlich wird Eigentum als eine Sache, die einer Person gehört, aufgefasst. Nach einem wissenschaftlichen Verständnis besitzt eine Person aber nicht eine Sache als solche, sondern bloß verschiedene Rechte an ihr. So kann man Eigentum als eine weitreichende soziale Beziehung unter Menschen deuten, die die Verfügungsmöglichkeiten über physische oder gesellschaftlich konstruierte Dinge gestattet oder beschränkt (vgl. Elwert 1999: 1143; Hann 1998: 2f.). Es ist also nicht der Anspruch auf eine Sache entscheidend, sondern die Anerkennung des Anspruchs durch andere, d.h. die Gesellschaft. Die Verfügungsmöglichkeiten haben die Gestalt eines Rechtebündels. Dieses Bündel lässt sich in vier Arten von Rechten aufteilen:

[1] Zum Urheberrecht existieren zwei Denkschulen, das kontinentaleuropäische Autorenrecht und das Angloamerikanische Copyright. Diese sind weitgehend gleich, durch eine unterschiedliche Auffassung der Rolle des Autors existieren im Autorenrecht aber zusätzlich moralische Rechte, etwa das, auch nach einem Verkauf die Verwendung des Werkes zu beschränken. Zudem ist anzumerken, dass Urheberrecht in verschiedenen Staaten unterschiedliche ausgestaltet ist (vgl. Grassmuck 2002: 51ff.).

Rechte des Gebrauchs, Rechte der Veränderung, Rechte der Übertragung und die Kompetenzkompetenz (Richter/Furubotn 1996: 82).[2] Die Eigentumsrechte können durch Einbettung stabilisiert werden, beispielsweise in Form staatlicher Gesetzgebung oder religiöser Bestimmungen (Elwert 1999: 1143).

Der Anspruch auf bestimmte Verfügungsrechte kann durch unterschiedliche ideologische Erklärungen legitimiert werden, beispielsweise die Vorstellung, dass Eigentum durch Arbeit geschaffen wird (Nuss 2002: 18f.). Vor der Aneignung einer Sache muss diese erstmal eigentumsfähig sein, d.h. sie muss klar abgegrenzt und spezifiziert sein (vgl. Richter/Furubotn 1996: 100). Wer bei der Aneignung welche Verfügungsrechte bekommt, ist durch gesellschaftliche Konvention geregelt.

Die Gestalt des Bündels der Verfügungsrechte variiert in jeder Gesellschaft. So mag zum Beispiel das Recht der Übertragung bezüglich bestimmter Sache nicht existieren. Die gesellschaftlichen Konventionen regeln auch, wie Verfügungsrechte übertragen werden können, etwa in welcher Weise sie vererbt werden können und ob Verträge darüber geschlossen werden können. Die Möglichkeit von Vertragsabschlüssen macht die Existenz einer Instanz der Überwachung ihrer Einhaltung notwendig. Verträge müssen überwacht werden können und im Falle eines Vertragsbruchs muss eine Sanktionierung erfolgen können. Wenn dies nicht der Fall ist, wird es zu einem Vertrauensverlust kommen.

Der Inhaber der Verfügungsrechte kann verschiedener Gestalt sein, es kann sich um eine Einzelperson handeln (exklusives Privateigentum), eine Gruppe (exklusives Kollektiveigentum), oder auch jeden (offenes Kollektiveigentum). Die Kompetenzkompetenzinhaber entscheiden über die weitere Verteilung der Rechte und können einen Teil ihrer Rechte an andere übertragen. Welche Rechte das sind, hängt auch von der Gestalt der Sache ab: es gibt Dinge, die nach einmaligen Gebrauch verbraucht sind; andere sind für eine mehrmalige Nutzung vorgesehen, diese können dabei aber mit der Zeit ihre Brauchbarkeit verringern (oder eben auch nicht). Bei immateriellen Dingen kommt als Gegenstandscharakteristikum hinzu, dass sie nicht nur nicht abnutzbar sind, sondern auch von verschiedenen Menschen ohne gegenseitige Beeinträchtigung gleichzeitig an verschiedenen Orten genutzt werden können. Bei der Übertragung von Verfügungsrechten kann das ganze Bündel (inklusive der Kompetenzkompetenz) weitergegeben werden, oder auch nur einzelne Rechte. Hierbei entschiedet der Kompetenzkompetenzinhaber, welche dies sind. Er hat auch die Möglichkeit an unterschiedliche Personen verschiedene Bündel weiterzugeben, in denen einzelne Rechte identisch sind. Er kann bestimmen, ob Rechte an eine bestimmte Person vergeben werden, oder auch von einer unbestimmten Gruppen genutzt werden können.

[2] Richter/Furubotn beachten das Konzept der Kompetenzkompetenz nicht. Der Begriff kommt aus dem Staatsrecht und bezieht sich auf die Kompetenz (das Recht) über Kompetenzen (Rechte) zu verfügen oder neue zu begründen (Streinz 2001: 49 [RN 121]).

4.1.2 Eigentum bezüglich Freier Software

Software ist eine immaterielle Sache; die Daten (Inhalt) können auf verschiedenartigen Medien (Träger) gespeichert werden, d.h. es ist irrelevant, ob die Software sich auf einer Diskette, einem Laptop oder auf einem Internetserver befindet, sie bleibt gleich. Daher bezeichnet Sabine Nuss Software auch als ein Informationsprodukt (Nuss 2002: 11). Im Gegensatz zu materiellen Dingen, nutzen sich Informationsprodukte durch den Gebrauch nicht ab. Außerdem kann Software aufgrund ihrer digitalen Form (fast) ohne Kosten unendlich vervielfältigt werden (ebd.: 11). Wie sieht es daher mit den Möglichkeiten zur Gestaltung der Verfügungsrechtsstruktur von Software aus?

Die Verfügungsrechte von Software sind durch Verträge, in Gestalt von Lizenzen, geregelt. Die Lizenzen bauen auf dem Urheberrecht auf. Der Autor der Software ist zunächst der Eigentümer und Rechteinhaber, wenn er stirbt gehen sie für eine bestimmte Zeit auf seine Erben über.[3] Danach wird die Software zu Public Domain, d.h. zu Allgemeingut. Der Copyrightinhaber kann seine Rechte durch Vertrag an andere abgeben. Der Arbeitsvertrag eines Angestellten kann beispielsweise schon vor der Entstehung der Software festlegen, dass die Rechte an der Software, die er für seinen Arbeitgeber schreiben wird, dem Arbeitgeber zufallen (um genau dem vorzubeugen hat auch Richard Stallman seine Anstellung am MIT zu Beginn des GNU Projekts gekündigt. Vgl. 3.2.1).

Der Copyrightinhaber kann über die weitere Verwendung der Software entscheiden, er hat die Kompetenzkompetenz. Entweder er tritt sein Copyright beispielsweise durch Verkauf an jemand anderes ab (inklusive der Kompetenzkompetenz) oder er gibt nur einzelne Verfügungsrechte weiter, das geschieht im allgemeinen durch eine Lizensierung an andere. An diesem Punkt unterscheidet sich Freie Software von anderen Typen der Softwarelizensierung. Microsoft gibt beispielsweise, als Vertreter proprietärer Lizenzpolitik, in der Lizenz für Word 97, das mit dem Kauf eines PCs erworben wurde, dem Käufer folgende Rechte: eine Kopie der Software zu installieren und zu benutzen; bei der Nutzung mittels eines Netzwerks muss für jeden weiteren Rechner eine eigene Lizenz erworben werden; die Erstellung einer Sicherungskopie allein zu Archivierungszwecken; die einzige Möglichkeit der Übertragung ist die des Verkaufs des Rechners mit dem die Software erworben wurde, wenn der Käufer der Lizenz zustimmt; die Lizenz gilt nur für den mit der Software gekauften Rechner (Microsoft 1995). Alles was darüber hinaus geht ist untersagt. Der Lizenzerwerber erhält also eingeschränkte Nutzungsrechte, keinerlei Rechte der Veränderung und ein einziges der Übertragung, d.h. der Copyrightinhaber gibt bei dieser Lizenz nur sehr wenige Rechte weiter.

Die Lizenzen Freier Software haben dagegen eine deutlich andere Ausgestaltung

[3] Die Zeitspanne ist je nach nationaler Gesetzgebung verschieden.

bezüglich der Verfügungsrechte. Diese orientieren sich an den von Richard Stallman geforderten Freiheiten für Freie Software (vgl. 3.2.1): bezüglich der Nutzung zu jedem Zweck, der Veränderung, der Verteilung von Kopien und der Verteilung von modifizierten Versionen. Diese Freiheiten bzw. Rechte sind die Bedingung dafür, das eine Software als *Freie* Software gilt. Die GPL,[4] wie auch die BSD Lizenzen,[5] gewähren sie. Interessant ist bei Lizenzen Freier Software allerdings, wie die Kompetenzkompetenz geregelt ist (vgl. Free Software Foundation 2003a; Regents of the University of California 1993): Die BSD Lizenzen geben fast alle Kompetenzrechte weiter, allerdings muss der Copyrightvermerk der Universität Berkeley erhalten bleiben. Ansonsten darf bei einer Weitergabe der Software (in der ursprünglichen oder modifizierten Form) eine neue Lizenz gewählt werden – auch eine proprietäre. Bei der GPL verbleibt die Kompetenzkompetenz dagegen – bedingt durch das Copyleft – völlig beim ursprünglichen Copyrightinhaber: eine unter der GPL veröffentlichte Software oder Teile von ihr müssen ebenfalls unter der GPL veröffentlicht werden. Allein der Copyrightinhaber hat das Recht, die Software unter eine andere Lizenz zu stellen. Hierbei verbleibt der bereits unter der GPL veröffentlichte Code unter der GPL, d.h. eine Veröffentlichung unter der GPL ist auch für den Copyrightinhaber nicht mehr zurücknehmbar. Bei einer Veröffentlichung unter einer weiteren Lizenz (duale Lizensierung) kann die Software *zusätzlich* unter eine proprietäre Lizenz gestellt werden. So können alle weiteren Entwicklungen unter der proprietären Lizenz auch proprietär bleiben, solange sie vom Copyrightinhaber nicht in die unter der GPL veröffentlichten Version eingefügt werden.[6]

Neben den BSD Lizenzen und der GPL gibt es noch eine Reihe von weiteren Lizenzen Freier Software, wobei die hier besprochenen wohl die verbreitetsten sind (vgl. Open Source Technology Group 2004).[7] Die Lizenzen Freier Software lassen sich in zwei Kategorien einteilen: solche mit und solche ohne Copyleft. An diese Listen orientieren sich auch andere Akteure in diesem Bereich (vgl. Gentoo Foundation 2001-2004; Open Source Technology Group 2004). Es fragt sich, ob es eine

[4] http://www.fsf.org/licenses/gpl.html

[5] Es gibt sowohl die *Original BSD License*, als auch die modifizierte. Sie unterscheiden sich darin, dass in der Original License, vorgeschrieben ist, dass bei jeder Werbung der Hinweis enthalten sein muss, dass die Software teilweise Code der Universität Berkeley und anderer Beitragender enthält. Beide sind unter http://www.xfree86.org/3.3.6/copyright2.html#5 zu finden.

[6] Duale Lizensierung ist nicht ungewöhnlich. Sie kann für Unternehmen oder einzelnen Personen sinnvoll sein, wenn sie ihre Software unter einer freien Lizenz veröffentlichen möchten, aber zusätzlich die wirtschaftlichen Vorteile einer proprietären Lizenz, z.B die Einnahme von Lizenzgebühren, nutzen möchten. Nutzer können die proprietäre Lizenz bevorzugen, wenn sie ihrerseits Software, die auf der ursprünglichen basiert, mit einer proprietären Lizenz veröffentlichen möchten. Bei einer dualen Lizensierung muss der Nutzer entscheiden, unter welcher Lizenz er die Software nutzen möchte und die dann vorgeschriebenen Lizenzvorschriften einhalten, tut er das nicht, begeht er einen Vertragsbruch.

[7] Unter der *GNU Lesser General Public License* (LGPL) sind zwar mehr Projekte veröffentlicht, sie soll hier aber wegen ihrer großen inhaltlichen Nähe zur GPL nicht weiter besprochen werden. Der große Unterschied ist, dass sich die LGPL den Copylefteffekt für Software, die auf Bibliotheken basieren, die wiederum unter der LGPL veröffentlicht sind, außer Kraft setzt (vgl. Free Software Foundation 2002a).

Übereinstimmung der unterschiedlichen Lizenzmodelle mit den jeweils verschieden Ursprüngen der Projekte gibt. Abschließend soll noch eine Anmerkung zu der kommerziellen Nutzung von Freier Software gemacht werden: Freie Software heißt nicht, dass diese kostenlos sein muss. Nur wenn das explizit in der Lizenz vorgeschrieben ist, ist es der Fall – dies trifft auf die GPL trotz manch anderer Auffassung nicht zu (vgl. Free Software Foundation 2003b).

Außer Freier und proprietäter Software gibt es noch weitere Typen von Softwarelizensierungen: Public Domain, semi-freie Software, Free- und Shareware. Public Domain Software ist ohne jeden Copyrightvermerk, d.h. jeder kann sie nutzen wie er will. In Deutschland ist es nicht möglich Software, oder andere „Schriftgüter“ unter Public Domain zu veröffentlichen, da hier der Autor immer bestimmte Rechte an seinem Werk behält (ob er will oder nicht). Erst mit auslaufen des Copyrights fällt ein Werk in die Public Domain. In den Vereinigten Staaten, ist es dagegen möglich ein neu geschaffenes Werk als Public Domain zu deklarieren. Public Domain unterscheidet sich von Freier Software durch das Fehlen des Copyright, außerdem ist nicht immer der Quellcode beigelegt. Bezüglich der Verfügungsrechte bedeutet dies, dass der Nutzer alle Rechte der Nutzung, der Veränderung und der Weitergabe hat, unabhängig davon, ob es technisch möglich ist. Er erhält auch die Kompetenzkompetenz, mit Ausnahme, dass er nicht auf exakt den gleichen Code ein Copyright beanspruchen kann, sehr wohl aber auf Veränderungen. Semi-freie Software unterscheidet sich von Freier Software durch ein Verbot der kommerziellen Nutzung, d.h. hier ist Recht der Nutzung nur unter bestimmten Bedingungen erlaubt. Freeware ist proprietäre Software, die allerdings frei verteilt und ohne Nutzungsgebühr verwendet werden darf. Bei Shareware ist ebenfalls die Weitergabe erlaubt, aber nach einer gewissen Zeit der Nutzung ist man verpflichtet eine Gebühr zu zahlen. Bei Free- und Shareware hat der Nutzer jedes Recht der Weitergabe, keines der Veränderung und keinerlei Kompetenzkompetenz. Allerdings hat er bei Freeware alle Nutzungsrechte, währender diese bei Shareware nur behält, indem er nach der vorgegebenen Zeit eine Gebühr zahlt (Free Software Foundation 2002b).

4.1.3 Fazit

Die beschriebenen Lizenzen grenzen das Feld der Freien Software auf eine objektive Weise ab. Auf der Grundlage existierender Bestimmungen zum Urheberrecht definieren sie ein spezifisches Bündel von Rechten und unterscheiden Freie Software damit von Software unter anderen Lizenzen. Allerdings ist damit der sozialen Formation der Entwickler Freier Software lediglich ein äußerer Rahmen gesetzt. Über die in diesem Feld Arbeitenden ist noch nichts ausgesagt. Im nächsten Abschnitt wird untersucht, wie die Entwickler diese objektiven Grenzen von innen ergänzen.

4.2 Subjektive Abgrenzung nach außen

4.2.1 Selbstverständnis

Neben der äußeren, rechtlichen gibt es auch eine subjektive Abgrenzung. Dabei geht es zum einen um die Unterscheidung von Dazugehörigen und Außenstehenden. Zum anderen gehört dazu die kollektive Identität der Entwickler. Im folgenden Abschnitt werden diese beiden Teilbereiche der subjektiven Abgrenzung beschrieben.

Es gibt gewisse Klischees über den Lebensstil von „Computerfreaks", zu denen auch die Entwickler Freier Software gehören: Zunächst sind „Computerfreaks" männlich, wenn sie noch Jugendliche sind, tragen sie eine Brille, haben Pickel und tragen unmodische Kleidung, wenn sie älter sind tragen sie immer noch eine Brille, haben keine Pickel mehr, aber womöglich lange Haare und einen Bart und tragen Jeans und T-Shirt (letztlich immer noch unmodische Kleidung). Sie sind nachtaktiv, koffeinabhängig und leben in einem unaufgeräumten Zimmer, in dem neben allen möglichen technischen Geräten und Teilen, alte Essensreste, überfüllte Aschenbecher,[8] Kleidung usw. ein großes Chaos bilden. Ihr Lebensmittelpunkt ist der Computer. Ihre soziale Kompetenz ist rudimentär ausgeprägt, sie haben nur wenige Freunde, die ebenfalls Computerfreaks sind. Frauen sind für sie nur ein ferner Wunschtraum (vgl. Weizenbaum 1978: 160ff.; Helmers 1994). Wie weit diese Klischees mit der Wirklichkeit übereinstimmen, ist fraglich.[9] Zutreffend ist, dass Frauen einen verschwindenden Anteil ausmachen, 1,1 % sind nach den Ergebnissen der FLOSS-Studie (Ghosh et al. 2002:8) weiblich, nach meiner Umfrage sind es 2,2 %. Nicht zutreffend ist das Singledasein, nach der FLOSS Studie haben 58,5 % (Tab. 1)[10] eine Partnerschaft (ebd.: 11), nach meiner Umfrage sind es 50 % (Tab. 3). Aus Texten über Linus Torvalds und Richard Stallman geht hervor, dass sie längere Zeiträume fast nur vor dem Computer verbrachten (Torvalds/Diamond 2001: 23ff.; Moody 2001: 32).

Unabhängig von der Realität der Klischees, gibt es aber immer wieder Situationen, in denen die Entwickler sich selbst darauf beziehen. So wurde ich oft auf Personen hingewiesen, die dem dem Bild eines wahren *Geeks* oder *Nerds* entsprachen[11]

[8] Beim Thema Rauchen teilt sich das Klischee in zwei Meinungen auf, entweder ist der Computerfreak absoluter Nichtraucher, denn der Rauch und die verstreute Asche schadet nur der Hardware, oder er ist ein ausgemachter Kettenraucher, der ohne seine kontinuierliche Nikotinzufuhr nicht leben bzw. arbeiten kann.

[9] Bei meinen Beobachtungen auf den Linuxtagen ließen sich einzelne Personen ausmachen, die diesem Klischee entsprechen könnten, der Großteil tat es auf den ersten Blick nicht. Steven Levy schrieb zu dem Thema 1984: „„Though some in the field used the term *hacker* as a form of derision, implying that hackers were either nerdy social outcasts or *unprofessional* programmers who wrote dirty code, I found them quite different." (Levy 2001: 7).

[10] Die Nummern beziehen sich auf die Tabellen in Anhang A

[11] Zu der Bezeichnung *Nerd* steht im Jargon File: „1. [mainstream slang] Pejorative applied to anyone with an above-average IQ and few gifts at small talk and ordinary social rituals. 2. [jargon] Term of praise applied (in conscious ironic reference to sense 1) to someone who knows what's really important and interesting and doesn't care to be distracted by trivial chatter and silly status games.

oder es wurden abfällige Kommentare bezüglich Personen geäußert, die beim Social Event des Linuxtages allein an ihrem Laptop saßen, anstatt sich mit den anderen zu unterhalten. Zusätzlich gibt es recht viele Witze über die Schüchternheit gegenüber Frauen. Die Bezeichnungen *Geek* oder *Nerd* sind heute aus emischer Sicht positiv belegt (Raymond 2001). So gibt es auf der LinuxWorldExpo einmal im Jahr eine Quizshow, *The Golden Pinguin Bowl*, in der das Team der *Geek* gegen das der *Nerds* antritt (Delio 2001; Linuxworldexpo 2004). Neben den äußerlichen und sozialen Elemente zeichnen sich Geeks oder Nerds auch durch ein großes Wissen über Computer aus.

Allgemein ist ein gewisser Wissenstand die Voraussetzung und notwendige Bedingung, um überhaupt dazugehören zu können. Ohne Kenntnisse über Computer (Hardware), Software und bedingt über Programmierung,[12] ist eine Teilnahme nicht möglich. Diese auf Wissen basierende Grenzziehung ist auch den Entwicklern bewußt, wie es besonders ausgeprägt am folgenden Spruch ablesbar ist: "There are 10 types of people in the world: Those who understand binary and those who don't".[13] Hier wird die Teilung der Menschheit in zwei Gruppen vorgenommen. Binaries fungieren in diesem Beispiel als Symbol des (Computer)Wissens. Auch werden unwissende, nicht lernbereite Computerbenutzer gerne als *Dummies*, *Luser* (die Verbindung von loser und user) oder *DAUs* (dümmster anzunehmender User) bezeichnet. Bei diesen Beurteilungen zählt weniger der aktuelle Wissensstand, als viel mehr die Bereitschaft der jeweiligen Person zum Denken und zum offenen Umgang mit Neuem. Es wird von einer Grundintelligenz der Entwickler wie auch der Benutzer ausgegangen.

> „'It doesn't work.' Give the programmer some credit for basic intelligence: if the program really didn't work at all, they would probably have noticed." (Tatham: 1999).
>
> „Anscheinend ist heutzutage in der Hilfsliteratur für Computer eine der Hauptthesen ‚Nehmen Sie an, daß der Benutzer den IQ einer Kartoffel besitzt'. Dem werden wir sicherlich nicht folgen." (Lyx-Team 2002)

Unter den Entwicklern scheinen äußerliche Attribute einer Person, wie Alter, Status (Titel) oder Aussehen, erst einmal irrelevant zu sein, so kommen beispielsweise auch Äußerungen wie „unser bester C-Programmierer war 13 Jahre alt" zustande.[14] Dieser Umstand lässt sich nur bedingt mit der virtuellen Umgebung der Kooperation

Compare geek"(Raymond 2001). Und zur Bezeichnung *Geek* lässt sich folgendes finden: „A person who has chosen concentration rather than conformity; one who pursues skill (especially technical skill) and imagination, not mainstream social acceptance."(ebd.).

[12] Hierbei muss bedacht werden, dass ein Teil der Entwickler sich mit Übersetzungen und der Dokumentation beschäftigt. Tätigkeiten, für die (wenn überhaupt) nur rudimentäre Programmierungskenntnisse notwendig sind.

[13] „10" bedeutet im binärem Zählsystem „2". Gesehen auf T-Shirt.

[14] Interview mit einem Mitarbeiter von WorldForge am 12.7.2003 in Karlsruhe.

erklären, in der es zunächst nicht möglich ist, andere und ihre äußere Erscheinung zu sehen, da sich bereits unter den Hacker am MIT Kinder oder Jugendliche befanden (siehe Kap. 3.1.1). Es stellt sich allerdings die Frage, ob eine andere Art der Attribute zählt, beispielsweise eine vorhandene Nerdhaftigkeit oder ob Frauen zunächst einmal skeptischer betrachtet werden.[15]

Statements auf Tassen oder T-Shirt geben das Selbstbild der Entwickler und ihre Beziehung zu den Außenstehenden wieder. Sie können die Form eines Koffeinmoleküls haben, aber auch Statements wie „There's no place like 127.0.0.1“, „Root, „Go away or I will replace you with a very small shell script“, „No I will not fix your Computer“ oder „PEBKAC“ sein.[16] Verstärkt wird die Abgrenzung durch eine eigene Sprache: Viele Begriffe stammen noch aus dem MIT Jargon (vgl. Kap. 3.1.1), aber es werden auch viele Unixbegriffe, insbesondere Kommandos, im normalen Sprachgebrauch verwendet, beispielsweise „grep“ für durchsuchen.

Die oben beschriebenen Punkte haben Entwickler Freier Software zu großen Teilen auch mit andern Personen gemeinsam, die sich intensiv mit Computern beschäftigen. Von ihnen unterscheiden sie sich aber durch ein bestimmtes Set an Werten und Normen. Nicht nur technisches Wissen ist notwendig, sondern auch kulturelles, um dazu zu gehören.

Dieses kulturelle Wissen bezieht sich auf den Code selbst, die Tätigkeit des Entwickelns, auf die Form der Kooperation untereinander und die politische Dimension Freier Software. Zunächst ist ein Einverständnis darüber vorhanden, dass Software frei sein sollte. Der Hintergrund dieser Überzeugung kann allerdings verschieden sein (siehe Kap. 4.3.1). Anders als bei Unternehmen, die Software durch den Verkauf bzw. Lizensierung als eine Möglichkeit Profit zu erzielen ansehen, steht bei Freier Software, die Software als solche im Mittelpunkt.[17] Daher sind die Rahmenbedingungen für Freie Software auch anders gestaltet: Aspekte wie Marketing, um von einer Version möglichst viele Lizenzen verkaufen zu können, spielen eine untergeordnete Rolle. Das wirkt sich beispielsweise auf den Zeitpunkt aus, zu dem eine Version veröffentlicht wird. Bei Freier Software soll dies erst geschehen, wenn die Entwickler den Entwicklungsstand für geeignet erachten. Das Linux Projekt nimmt hier eine Zweiteilung vor: die Versionen mit den ungeraden Nummern (z. B. 2.*3*.n) haben die neuesten Features eingebaut, die aber nicht immer sehr stabil sind. Diese Versionen

[15] Mir wurde von einem Fall berichtet, bei dem eine Entwicklerin einen neutralen Namen benutzte, da sie vorher schlechte Erfahrungen aufgrund ihres Geschlechts gemacht hat. Ob diese Erfahrung verallgemeinerbar ist, ist unklar.

[16] „127.0.0.1“ ist die Adresse von Home (das persönliche Verzeichnis im einem Unix-System); „Root“ ist ein Benutzer mit sämtlichen Rechten (häufig der Systemadministrator); „Shell Scripts“ sind kleine Programme zur Erledigung meist trivialer Aufgaben; „PEBKAC“ bedeutet „Problem Exists Between Keyboard and Chair“. Weitere T-Shirts sind unter anderem unter http://www.thinkgeek.com/tshirts/ zu finden.

[17] Profiterzielung vergleichbar proprietärer Software ist bei Freier nicht wirklich realistisch, da eine Bedingung für die Freiheit der Software die Möglichkeit diese weiter zu verteilen ist (vgl. Kap. 4.1.2). Zu anderweitigen wirtschaftlichen Nutzung Freier Software siehe Raymond 2000c.

richten sich an Mitentwickler und dienen dazu möglichst viele Personen an der weiteren Verbesserung des Codes teilhaben zu lassen. Die Versionen mit der geraden Nummerierung (z. B. 2.*4*.n) sind ausführlich getestet und laufen recht stabil, ihre Zielgruppe sind die (End)Benutzer (Kofler 2001: 41). Allerdings soll ein Programm nicht nur irgendwie funktionieren, sondern es sollte möglichst auch gut geschrieben sein.[18] Es gibt Vorstellungen, wie Code zu sein hat: elegant, im Sinne von "Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away" (Raymond 2000a) – eine Vorstellung, die schon bei den Hackern am MIT vorherrschte (vgl. Kap. 3.1.1). Jon „Maddog" Hall[19] umschreibt es ein wenig blumiger: „If software is elegant then the world smiles, the sun comes out, the clouds are fluffy and everything else."[20] Unter Umständen ist aber auch *quick & dirty*-Code anerkannt, wenn eine Lösung schnell gebraucht wird. *Quick & dirty*-Code wird aber nur als Übergangslösung angesehen. Andere Entwickler werden anders als im kommerziellen Sektor nicht als Konkurrenz oder Feinde angesehen, sondern als potentielle Mitentwickler. So ist die Einstellung, dass man ohne die anderen nichts wäre, weit verbreitet. Linus Torvalds äußert sich zu folgt zu diesem Punkt: „Als ich Linux erstmals ins Internet stellte, hatte ich das Gefühl, in die jahrhundertealten Fußstapfen der Wissenschaftler und Forscher zu treten, die ihre Arbeit auf den Grundfesten anderer aufsetzten – auf den Schultern von Giganten, um mit Isaac Newton zu sprechen." (Torvalds 2001: 103). Auch kritisches Feedback, insbesondere Fehlermeldungen, ist daher eindeutig erwünscht, da es zu einer Verbesserung führen und so die Qualität der Software steigern könnte. Anders als bei Unternehmen steht hier nicht die Kontrolle der Programmierer im Vordergrund und es erfolgt keine Statusreduzierung, sondern es ist zunächst einmal eine Möglichkeit zum Lernen – für alle. So schreibt Georg Greve, der Präsident der FSF Europe, dass „Für das technische Gelingen eines Projekts [...] die Rückmeldung von Fehlern ebenso wichtig [ist,] wie das Entwickeln selbst." Weiter weist er auf den Zusammenhang zwischen der schnellen Entwicklung junger Projekte und Fehlerrückmeldungen hin. So würden in Projekten qualitativ hochwertige Rückmeldungen ebenso hoch wie guter Code geschätzt werden (Greve 2004: 103f.). Wichtig ist auch Selbständigkeit, d.h. wenn einem etwas nicht gefällt, soll man selbst aktiv werden und kann nicht erwarten, dass andere das Problem beheben. Die Entwickler sehen ihre Leistung als einen freiwilligen Beitrag an, der sie außer zur Fehlerkorrektur erstmal zu nichts weiter verpflichtet.[21] Hinzu kommt, dass von den Beteiligten erwartet

[18] Aufgrund des freizugänglichen Quellcodes, sind alle hervorragenden Leistungen, aber auch Unachtsamkeiten nachzuvollziehen und einer bestimmten Person zurechenbar.

[19] Jon „Maddog" Hall ist seit 1969 im Computersektor tätig und ist seit 1995 Präsident von Linux International (Linux International 1994-2004). Man könnte ihn als den lieben Onkel bezeichnen und er genießt ein sehr hohes Ansehen – u.a. auch wegen seines offenen und hilfsbereiten Charakters.

[20] Interview am 24.6.2004 in Karlsruhe.

[21] Diese Einstellung ließ sich gut an einer Auseinandersetzung zwischen einem Unternehmen (4Front Technologies) und Mitarbeitern des Linux-Projekt beobachten. Siehe den Thread „Stop the Linux kernel madness" (http://marc.theaimsgroup.com/?t=108751764100001&r=5&w=2)

wird, bei einem Problem erst einmal selbst nachzudenken und sich im Zweifelsfalle selbst kundig zu machen. So kommt es auch zu dem Ausspruch „RTFM – Read The F Manual“[22].

Diese drei Themen – Code, Entwickeln, Kooperation – und ihr Bezug zueinander sind seit den 1990er Jahren von den Beteiligten verstärkt reflektiert worden. Hier ist besonders Eric Raymonds Essay "The Cathedral and the Bazaar" aus dem Jahre 1997 zu nennen.[23] Darin kommt er zu dem Schluss, dass die entscheidende Neuerung durch das Linux Projekt nicht auf der technischen, sondern auf der sozialen Ebene stattfand (Raymond 2000a). Er erkannte im Stil der Zusammenarbeit des Linux-Projekts ein dezentrales Basarmodell. Dagegen setzt er den herkömmlichen Entwicklungsstil mit seinen strengen Hierarchien und isolierten Arbeitsteams, den er mit einer Kathedrale vergleicht. Nachdem solch ein Team hochqualifizierter Programmierer seine Arbeit abgeschlossen hat und für soweit fehlerfrei hält, dass es den Nutzern zumutbar ist, wird es offiziell veröffentlicht. In der Zeit bis zum nächsten Update (meist mindestens ein halbes Jahr) versuchen sie, weitere Fehler zu beheben. Nach dieser kurzen Beschreibung der Kathedrale, wendet sich Raymond der ausführlichen Vorstellung des Basars zu. Dort schart sich um den Leiter eines Softwareprojekts (meist dessen Gründer) eine große Gruppe von Mitentwicklern und (Beta)Testern. Ihre Arbeitsweise ist ungefähr folgende: das Programm wird in einer rudimentären Version veröffentlicht; jemand findet ein Problem; der nächste versteht es; der dritte entwickelt eine Lösung dafür. Die Beteiligten müssen nicht einmal zum Projekt gehören, sondern sind vielleicht nur Nutzer, denen das Problem aufgefallen ist. Dadurch wächst die Zahl der Beitragenden enorm, und mit ihr die geistigen Ressourcen, die in das Projekt miteinfließen. Die Kommunikation findest über Mailinglisten, Newsgroups etc. statt. Raymond vergleicht diese Arbeitsteilung, in der er den Hauptunterschied zum Kathedralenmodell sieht, mit der sozialwissenschaftlichen Delphi-Methode.[24] Das Basarmodell zeichnet sich durch häufige Veröffentlichungen aus, damit die Entwickler von den Benutzern ständig wertvolle

insbesondere http://marc.theaimsgroup.com/?l=linux-kernel&m=108752076428626&w=2; http://marc.theaimsgroup.com/?l=linux-kernel&m=108752196412164&w=2; http://marc.theaimsgroup.com/?l=linux-kernel&m=108752195825184&w=2; http://marc.theaimsgroup.com/?l=linux-kernel&m=108752279900751&w=2; http://marc.theaimsgroup.com/?l=linux-kernel&m=108752279809091&w=2.

[22] Das „F“ wird je nach einzelnem Entwickler mit „fine“ oder „fucking“ übersetzt.

[23] Eric Raymond ist selbst Entwickler Freier Software. Er hat nie eine offizielle Ausbildung in diesem Bereich erhalten, sondern auf den Großrechner in der Firma seines Vaters sich seine Kenntnisse autodidaktisch erworben. Seit 1991 ist der der Herausgeber des „New Hacker's Dictionary“, auch der „Jargon File“ genannt (Oringel 1998). Außerdem engagierte er sich in den 1980er Jahren intensiv beim GNU-Projekt, so war er bis 1992 ein wichtiger Mitentwickler von Emacs (Williams 2002: 156).

[24] Die Delphi-Methode ist eine spezielle Form des Experteninterviews ist: Mehrere Experten werden getrennt interviewt, anschließend werden ihnen die anonymisierten Interviews der anderen vorgelegt. Daraufhin geben sie eine erneute Stellungnahme ab, welche wiederum an die anderen verteilt wird. Diese Prozedur wird so lange wiederholt, bis ein Konsens zwischen den Experten erreicht wird (Hillmann 1994: 142).

Fehlermeldungen bekommen können. Hierbei gilt der Satz, den Raymond das Linussche Gesetz getauft hat: „Given enough eyeballs, all bugs are shallow".[25] Auf diese Weise folgen die Veröffentlichungs-, Test- und Verbesserungsphasen sehr schnell aufeinander. Wichtig beim Basarmodell ist, dass der Projektleiter eine hohe soziale Kompetenz braucht, denn er muss seine Mitarbeiter ermutigen und es verstehen, sie durch Anerkennung zu belohnen. Zusätzlich muss er in er Lage sein, gute Ideen zu erkennen und sie integrieren können (Raymond 2000a).[26] Dieser Essay war für die Identitätsbildung der Entwickler Freier Software von entscheidender Bedeutung. Viele begannen, sich mit dem, was sie seit Jahren taten, auf eine neue Weise auseinanderzusetzten. Nach Glyn Moody war ihnen das vorher zwar latent bewusst, doch erst Raymonds Analyse führte es ihnen direkt vor Augen (Moody 2001: 212). Die Analyse von Raymond wird weitreichend von den anderen Entwicklern geteilt,[27] so dass man sie mittlerweile als allgemeines Kulturgut und damit als Bestandteil ihres Selbstverständnisses ansehen kann. Das Basarmodell wird heute (beinahe gleichwertig mit Stallmans Definition von Freier Software) als Merkmal Freier Software verstanden.

Neben dem geteilten Werten und Normen wird das Wir-Gefühl durch geteilte Traditionen verstärkt. Die heutigen Entwickler beziehen sich auf die Werte und Normen der Gruppen, die der Tradition zugeschrieben werden und orientieren sich an deren Handlungsweisen. Die gemeinsamen Wurzeln lassen sich im Computerbereich finden, beispielsweise die Hacker vom MIT, die das System (die Computer) erforschen und optimieren wollten, oder auch die frühen Unix-User, die sich durch eine enge und freie Kooperation auszeichneten – Unix ist heute wohl die verbreitetste Betriebssystemart innerhalb des Bereichs Freier Software. Jon „Maddog" Hall geht sogar noch weiter in die Geschichte zurück, zu Alan Turing, Grace Murray Hooper, Maurice Wilkes.[28] Neben den Wurzeln gehören auch „öffentliche Personen" und Geschichten zu den gemeinsamen Traditionen. Beispiele sind hier die Auseinandersetzung, die Richard Stallman Anfang der 1980er Jahre mit Mitarbeitern von Xerox hatte, da diese ihm nicht den Quellcode für einen fehlerhaften Druckertreiber geben wollten,

[25] Nach Linus Torvalds besagt das *Linussche Gesetz* allerdings etwas anderes, nämlich, „ das unser Motivationen in drei Kategorien fallen. Fortschritt wird durch das Erleben dieser Motivationen als ‚Phasen' in einem Entwicklungsprozess erreicht, bei dem es darum geht, von eine Kategorie zur nächsten zu gelangen. Diese Kategorien lauten ‚Überleben', ‚Sozialleben' und ‚Unterhaltung'." (Torvalds 2001b: 14)

[26] Es ist nicht ganz klar, wen Eric Raymond mit der Kathedrale assoziiert. Zunächst erscheint es so, dass er sich auf Unternehmen bezieht, aber wenn man dem Stallman-Biograph Sam Williams folgt, zielt er vielmehr auf das GNU Projekt (Williams 2002: 159).

[27] So bezogen sich einige meiner Gesprächspartner mehr oder weniger explizit auf Raymonds Analyse. Bei Nachfrage stellte sich heraus, dass sie ihr inhaltlich zustimmen, auch wenn Raymond als Person zumindest in Deutschland recht umstritten ist, u.a. wegen seines Faibles für Schusswaffen (vgl. Raymond 2004).

[28] Alan Turing, der Erfinder der Turingmaschine aus dem Jahre 1936, gilt als einer der Urväter der Computer. Als einer der legendären Dechiffrierer von Bletchley Park hat er den Enigma-Code mitgeknackt. Grace Murray Hooper gilt als die erste Programmiererin überhaupt. Maurice Wilkes entwickelte Ideen für fundamentale Innovationen in der Programmierung. Interview am 24.6.2004.

damit er die Fehler beheben konnte (vgl. Williams 2002: 1ff.; Grassmuck 2002: 222), oder auch der Disput zwischen Linus Torvalds und Andrew Tannenbaum in der Minix-Newsgroup zu Beginn des Linux Projekts (o.V. 1999).

Neben diesen strukturellen Analysen und eher philosophische Artikel[29] gibt es mittlerweile einige Texte, die sich mit der endogenen Geschichtsschreibung beschäftigen, teilweise in Form von Autobiographien (beispielsweise Torvalds 2001a), und die das kollektive Gedächtnis fördern.

4.2.2 Feindbilder

Neben der Selbstdefinition wird das Zusammengehörigkeitsgefühl durch die Identifizierung von Feindbildern gestärkt. Feindlich wird alles, was die Freiheit der Software bedroht, den Zugang zu Quellcodes versperrt (Copyrights, Patente, geheime Quellcodes), aber auch was Unselbstständigkeit fördert. Feindbilder sind insbesondere schon seit längerem Microsoft und seit 2003 die SCO-Group. Die SCO-Group hat sich durch eine Klage gegen IBM sehr unbeliebt gemacht, in der sie IBM Mitarbeitern die illegale Integration des AT&T Codes in den Linux Code vorwirft, und daher von Linux Benutzern Lizenzgebühren fordert. Allerdings verweigerte die SCO-Group lange einen Vergleich des AT&T-Codes mit dem Linux-Code und damit den Beweis, ob die Anschuldigungen überhaupt gerechtfertigt sein könnten. Bei den bisher durch geführten Überprüfungen kam es noch nicht zu einer Bestätigung der Vorwürfe. Verschärft wird die Situation dadurch, dass die SCO-Group aus dem Kauf von SCO durch Caldera, einem ehemaligen Linux-Distributor hervorgegangen ist (Vaughan-Nicols 2003; Chandar 2003; Bochers/Kuri 2003; Kuri 2004).

Interessant an beiden Fällen ist, dass sowohl Bill Gates, als auch die SCO-Group, als ursprünglich Dazugehörige betrachtet werden können, die sich später gegen die eigene Gruppe stellten. Bill Gates, der in seinen jungen Jahren ein Nerd war (vgl. Hagen 2002: 29ff.; Baumgärtel 2002: 114ff.), vollzog die Trennung bereits 1976 mit seinem "Open Letter to the Hobbyists" (siehe Kap. 3.1.3). SCO-Group wandelte sich vom Linux-Distributor zum rigorosen Verteidiger von proprietären Urheberrechten. Beide Fälle erfüllen also den Tatbestand des Verrats und stellen somit mehr als nur eine andere Auffassung über den Sinn und Zweck von Urheberrechten dar. Unternehmen, die proprietäre Software herstellen, machen sich noch nicht zum regelrechten Feind; ihr Tun wird als falsch empfunden, aber nicht notwendig als feindlich. Kommerzielle Aktivität ist überhaupt kein Problem: So hat es IBM, die mit Hardware, proprietäre Software, aber auch mit Freier Software Geschäfte machen, es durch seine Unterstützung für die Entwicklung Freier Software sogar geschafft, vom überlieferten Feind Nummer 1 (vgl. Young 1989 : 381ff.; Levy 2001: 41f.) zum

[29] Einige lassen sich auf der Seite der Free Software Foundation finden. Siehe http://www.fsf.org/philosophy/.

angesehenen Verbündeten zu bringen. Auch die als politisch eher radikal geltende FSF unterstreicht (Free Software Foundation 2004a), dass der kommerzielle Handel mit Freier Software zulässig und vollständig akzeptabel ist.

4.2.3 Fazit

Die beschriebenen Elemente der Selbstdefinition der Entwickler Freier Software zeigen, dass kollektive Wahrnehmungen und bestimmte Normen und Werte das Feld prägen. Damit entsteht eine subjektive Abgrenzung des Feldes nach außen. Die Zugehörigen teilen Einstellungen, Ansichten und Verhaltensnormen. Abgesichert und verstärkt werden diese durch einen Vorrat an gemeinsamen Geschichten und Symbolen. Gleichzeitig entstehen so klare Vorstellungen über die nicht zum Feld gehörenden, die sich in Feindbildern zuspitzen, welche die deutliche Trennung zwischen innen und außen versinnbildlichen. Es wird auch deutlich, dass das Selbstverständnis wichtiger ist, als die Definition des Anderen, wie der ausgeprägte Selbstreflexionsprozess und die Veränderbarkeit der Feindbilder belegen.

4.3 Interne Abgrenzungen

4.3.1 Open Source

Als Folge der durch Raymonds Essay „The Cathedral and the Bazaar“ angestoßenen Identitätsfindung, kam es zu einer inneren politischen Fraktionierung des Feldes. Im Frühjahr 1997 traf sich eine Gruppe bestehend aus wichtigen Beteiligten im Bereich Freier Software,[30] um dem Phänomen einen neuen Namen zu geben – *Open Source* (DiBona/Ockman/Stone 1999: 3).[31] Sie wollten damit zum einen die im Englischen problematische Doppeldeutigkeit des *Free* in *Free Software* umgehen[32] und sich zum anderen von der FSF und deren angeblicher antikommerzieller Einstellung distanzieren. Sie hofften, so *Open Source* in neuen wirtschaftlichen und weiteren gesellschaftlichen Bereichen zu verbreiten. Inhaltlich unterscheidet sich die *Open Source Definition*[33] eigentlich nicht von Stallmans Definition von *Freier Software*. Die Argumentation dieser *Open Source Initiative* (OSI) ist, anders als die der FSF, aber keine ethische, sondern eine pragmatische. Während Stallman die negativen gesellschaftlichen Folgen von proprietärer Software hervorhebt und mit Freier Software

[30] Zu dieser Gruppe gehörten u. a. Eric Raymond, Tim O'Reilly (Computerbuchverleger), Larry Austin (Präsident von VA Research – einem Hardwarehersteller) (DiBona/Ockman/Stone 1999: 3).

[31] Open Source ist die englische Bezeichnung für einen offenen (frei zugänglichen) Quellcode.

[32] *Free* wird eher mit gratis als mit unbeschränkt (Freiheit) assoziiert.

[33] http://opensource.org/docs/definition.php

ein Gegenmodell anbietet, betonen die Anhänger von *Open Source* die ökonomischen Vorteile und argumentieren mit der Effizienz und Qualität des Basar-Modells, das häufig auch als *Open Source-Methode* bezeichnet wird. *Open Source* kann man letztlich als eine Marketingstrategie verstehen (vgl. Raymond 1999b: 212).

Die FSF lehnt den Begriff und das dahinterstehenden Konzept *Open Source* ab, da er letztlich kontraproduktiv sei. Er würde letztlich nicht zu einem gesellschaftlichen Wandel führen, da Unternehmen und einzelne Personen aus pragmatischen (ökonomischen) Gründen zu *Open Source* wechselten und wenn es sich ergeben würde, aus der gleichen Motivation (ökonomische Erwägungen) heraus auch wieder auf proprietärer Software umsteigen könnten. Der Umstieg auf *Freie Software* basierend auf ethische Überzeugung sei dagegen nachhaltiger. Weiter weist die FSF daraufhin, dass auch der Begriff *Open Source* missverständlich ist, so würde die Betonung des freizugänglichen Quellcodes die weiteren Komponenten *Freier Software* (Rechte der Veränderung und Rechte der Weitergabe) verdecken, so dass als Folge einige Unternehmen absichtlich suggerieren, dass sie ihre Produkte als *Open Source* lizensierten, auch wenn sie de facto nur eine Einsicht in den Quellcode gewähren (Free Software Foundation 2004b).

Mittlerweile haben sich die Anhänger von *Freier Software* und *Open Source* zu zwei gefestigten Lagern entwickelt. Ein Pamphlet der FSF drückt es folgendermaßen aus: „The Free Software movement and the Open Source movement are today seperate movements with different views and goals, altough we can and do work together on some practial projects. [...] We disagree on basic principles, but agree more or less on the pratical recommendations.“ (Free Software Foundation 2004b). Allerdings erscheint es so, dass ein sehr großer Teil der Entwickler sich zwar aus irgendwelchen Gründen für eine der beiden Bezeichnungen entscheidet, den Hintergründen der Begriffe recht indifferent gegenüber steht.[34]

4.3.2 Hurd vs. Linux vs. BSD

Das Feld ist nicht nur in verschiedene politische Lager eingeteilt, die Entwickler distanzieren sich auch durch ihre Beteiligung an unterschiedlichen Projekten voneinander. Zum einen verläuft diese Trennung über ganze Systeme hinweg (z. B. Betriebssysteme), zum anderen aber auch auf der Ebene einzelner Programme (z. B. verschiedene Editoren).

Eine wichtige Trennlinie verläuft, wie bereits erwähnt, zwischen Linux und der verschiedenen BSD-Versionen. Diese beiden Systeme (hier ist zu bedenken, dass Linux

[34] Auch wenn in vielen Zeitungsartikeln und wissenschaftlichen Arbeiten zu dem Thema die Bezeichnung *Open Source* verwendet wird, wird in dieser Arbeit weiter von *Freier Software* die Rede sein. Das Verständnis von Freier Software als spezifische Eigentumsstruktur ist für diese Arbeit fundamental, pragmatische geschäftliche Erwägungen stehen dagegen nicht im Vordergrund.

nur ein Kernel ist, während die BSD-Versionen vollständige Betriebssysteme darstellen) sind wohl die verbreitetsten im Bereich Freier Software. BSD ist durch die früheren Rechtsstreite (vgl. Kap. 3.2.2) und den bis heute anhaltenden Hype um Linux seit Mitte/Ende der 1990er Jahre etwas in den Hintergrund gedrängt worden. So kam es, dass mich ein Mitarbeiter von NetBSD bei einem Interview zu Beginn erstmal darauf hinwies, dass nicht nur Linux Freie Software und BSD auch deutlich älter sei.[35] Ein anderes Beispiel ist die Beziehung zwischen den beiden Kerneln Linux und Hurd,[36] dem Kernel des GNU-Systems. Neben technischen Unterschieden[37] gibt es auch kulturelle, die die Wahl der Entwickler beeinflussen. Jedes bekanntere System oder Programm steht für bestimmte Attribute, beispielsweise technisch anspruchsvoll bei der Benutzung, Reduzierung auf das Nötigste, politische Aspekte (Grad der „Freiheit“ des Programms oder eine bestimmte Lizenz), Identifikation mit bekannten Entwicklern oder Nutzern[38]. Zwischen den Entwicklern von Programmen der gleichen Kategorie kann es zu einem freundschaftlichen Wettstreit kommen (beispielsweise *K Desktop Environment* (KDE)[39] vs. Gnome[40] oder auch Emacs[41] vs. vi[42]). Diese Konkurrenzsituation wird durch die Modulhaftigkeit von Unix und durch die Offenheit für neue Programme begünstigt.

Trotz dieser inneren Abgrenzung voneinander, sieht man sich als Teil eines größeren Ganzen. So ist es selbstverständlich, dass auf dem *Linux*tag auch BSD-Projekte vertreten sind. Außerdem beziehen sich beispielsweise auch BSDler auf Geschichten des Linux-Projekts.[43] Abschließend lässt sich feststellen, dass die Entwickler zwar die Unterschiede zwischen den Projekten pflegen und sich auch gegen ganz bestimmte abgrenzen, auf einer höheren Ebene betrachten sie sich aber als zusammengehörig.

4.3.3 Fazit

Innerhalb des Feldes Freier Software verlaufen also verschiedene Trennlinien. Dies deutet zunächst auf ein hohes Maß an Heterogenität hin. Allerdings ist festzustellen, dass die verschiedenen Teilfelder sich durchaus als ein Teil eines übergeordneten Feldes sehen. Die Konkurrenz zwischen Projekten hat eher sportlichen Charakter

[35] Interview am 5.4.2003 in Magdeburg.

[36] Das GNU Projekt sah Linux immer als suboptimale Übergangslösung an. Dies war nötig, da es Probleme und gravierende Verzögerungen bei der Entwicklung des eigenen Kernels gab. Die Version 0.0 von Hurd erschien erst 1996 (Raymond 1999a: 27f.; Stallman 2004; Bushell 1996).

[37] Die technische Ebene ist bei der Wahl eines Programms bzw. Projekts nicht bedeutungslos. ein Aspekt kann hierbei technisch-ästethische Erwägungen sein, d.h. jemand findet, dass eine bestimmte Komponente eine gute oder weniger gute Lösung ist. Dies ist eine ähnliche Ebene, wie die Eleganz von Code.

[38] Gespräche auf dem Linuxtag in Karlsruhe vom 7.-9.6.2002.

[39] http://kde.org

[40] http://gnome.org

[41] http://www.gnu.org/software/emacs/emacs.html

[42] http://www.bostic.com/vi

[43] Interview mit einem NetBSD-Mitarbeiter am 5.4.2003 in Magdeburg.

und ist wohl eine Folge des verbreiteten Wissens- und Perfektionsdranges. Eine Einschränkung ist hier jedoch zu machen: Zumindest aus Sicht der FSF ist die OSI Teil einer anderen *Bewegung*. Die Zusammenarbeit von Entwicklern aus beiden Lagern zeigt, dass ein hohes Maß an Übereinstimmung existiert. Als Bewegung, die auf eine klar benannte (politische) Veränderung hinarbeitet, wird die OSI von der FSF aber nicht akzeptiert. Diese unterschiedlichen politischen Ausrichtungen finden sich auch unter den Entwicklern.

4.4 Zusammenfassung

Entwickler Freier Software grenzen sich auf zwei Ebenen ab, nach außen und untereinander. Die Abgrenzung nach außen geschieht zum einen durch eine klar gezogene, objektive Ebene, die der Lizenzen. *Freie* Software unterscheidet sich von anderer Software (proprietäre, Public Domain, semi-freie, Free- und Shareware) durch die Gestaltung der verschiedenen Eigentumsrechte (Rechte der Nutzung, Rechte der Veränderung, Rechte der Weitergabe und die Kompetenzkompetenz). Die Verfügungsrechteweitergabe an die Benutzer geschieht über Lizenzen, d.h. der Urheber überträgt nicht sein vollständiges Eigentum an die untereinander gleichberechtigten Empfänger, sondern erlaubt ihnen gewisse Rechte in Anspruch zu nehmen. Die konkrete Ausgestaltung der übertragenen Rechte orientiert sich bei Freier Software an den von Richard Stallman definierten Bedingungen für Freie Software: das Recht der Nutzung zu jedem Zweck, das uneingeschränkte Recht der Weitergabe und das uneingeschränkte Recht der Veränderung. Die Kompetenzkompetenz bei den Lizenzen Freier Software ist allerdings unterschiedlich geregelt. Während bei der GPL die volle Kompetenzkompetenz beim Copyrightinhaber verbleibt, überlässt die BSD-Lizenz dem Empfänger auch diese Rechte, mit Ausnahme, dass die ursprünglichen Autoren genannt werden müssen. Der Rahmen dieser Lizenzen wird durch die staatliche Gesetzgebung, bestimmt, d.h. diese rechtliche Grenzziehung wird teilweise von außen mitgeprägt.

Die zweite Form der Abgrenzung nach außen ist auf der kulturellen Ebene angesiedelt. Neben dem mittlerweile positiv belegten Selbstbild als *Geek* oder *Nerd*, unterscheiden sich die Entwickler Freier Software – auch von anderen „Computerfreaks“ – durch ein bestimmtes Werte- und Normenset. Auffällig ist hierbei die Schlüsselfunktion des Wissens, die sich nicht ausschließlich auf den aktuellen Wissensstand einer Person bezieht, sondern auch auf die vorhandene aktive Lernbereitschaft. Diese Abgrenzung gegenüber Nichtdazugehörigen, wird durch äußere Symbole, aber auch einen eigenen Jargon bekräftigt. Außerdem gibt es einen Kanon von geteilten Vorstellungen bezüglich des Codes, der Tätigkeit des Entwickelns, der Kooperation untereinander und der politischen Dimension (die Freiheit der Software). Seit Mitte der 1990er Jahre hat ein kollektiver, innerer Reflexionsprozess darüber eingesetzt,

der sich in verschiedenen Schriften äußert und erheblich zur Wir-Gruppenbildung beträgt, u. a. durch den Bezug auf gemeinsame Traditionen und der daraus folgenden Pflege eines kollektiven Gedächtnisses. Schließlich wird die Unterscheidung zwischen Dazugehörigen und Außenstehenden durch die Identifizierung von Feinden, die sich explizit gegen die von den Entwickler geteilten Werte und Normen richten, versinnbildlicht.

Abgrenzungsprozesse finden aber nicht nur nach außen statt, sondern auch innerhalb des Feldes der Entwickler Freier Software. Zum einen verläuft diese Trennung anhand von ideologischen Einstellungen bzw. von Deutungen des Phänomens der Freien Software. Zum anderen distanzieren sich die Entwickler untereinander durch die Beteiligung an unterschiedlichen Projekten, die jeweils auch mit einer eigenen Projekt-Kultur verbunden werden und teilweise getrennte, lange Traditionen aufweisen (z. B. Linux vs. BSD). Trotz dieser internen Heterogenität, sehen sich die Beteiligten aber als Teil von etwas Ganzem.

Kapitel 5

Die Struktur des Feldes

In diesem Kapitel soll die Struktur des Feldes eingehender betrachtet werden. Dazu werden zunächst die für alle Entwicklungsprojekte gültigen räumlichen und kommunikativen Rahmenbedingungen dargestellt. Hierzu gehört auch ein Überblick über die technischen Mittel, mit denen die Entwickler diesen Bedingungen begegnen. Dann sollen anhand von Fallbeispielen konkrete Projekte kurz vorgestellt werden, um die Heterogenität des Feldes aufzuzeigen. Besonders berücksichtigt werden dabei die organisatorischen Unterschiede zwischen den Projekten, die als unterschiedliche politische Systeme verstanden werden können. Schließlich werden Grundlinien der Vernetzung der Projekte untereinander wie auch mit anderen Akteuren des Feldes aufgezeigt.

5.1 Räumliche Rahmenbedingungen und Infrastruktur

Der Raum des Feldes der Entwicklung Freier Software zerfällt in zahlreiche einzelne Projekte. Diese werden hier als Produktionsorte bezeichnet werden. Es handelt sich bei ihnen um virtuelle Orte innerhalb des Gesamtraumes des Feldes. Die konkrete Entwicklungsarbeit lokalisiert sich an zahlreichen physisch voneinander entfernten realen Orten, nämlich den Wohnzimmern oder Büros der einzelnen Entwickler.

Der virtuelle Raum, in dem die Projekte angesiedelt sind, beruht auf der Technologie des Internets. Das Internet basiert auf einer Kombination von Hardware (Server und Verbindungskabel oder -satelliten) und zum anderen auf Software (Protokolle, Server- und Clientprogramme). Die Internetdienste (www, mail, ftp, telnet, usw.) stellen die konkreten Bausteine dar, aus denen im virtuellen Raum einzelne Orte (z.B. Freie Software Projekte) konstruiert werden. Die Funktionsweise der jeweiligen Dienste und Programme geben den Gestaltungsspielraum vor. Das heißt, die Kommunikation im Raum ist nicht frei gestaltbar, sondern wird vom Regelwerk

der Internetdienste determiniert. Nur innerhalb dieser Grenzen ist die Gestaltung der Kommunikation und der darauf beruhenden sozialen Beziehungen möglich. Diese Determinanten bezeichnet Lawrence Lessig als Gesetz des Cyberspace (Lessig 2002: 224). Sie sind veränderbar, indem beispielsweise neue Internetdienste entwickelt werden, aber auch innerhalb der Programme, die Internetdienste für den Benutzer zugänglich machen (z.B. Browser oder Email-Programme), gibt es Variationsmöglichkeiten, d.h. die Autoren dieser Programme können den Raum selbst mitstrukturieren oder die Strukturen auf verschiedene Weise darstellen.

Innerhalb des so beschaffenen Rahmens kann jeder, der die technischen Fähigkeiten und Ressourcen hat, eigene Orte konstruieren, die aus Webseiten, Newsgroups, Chat-Channel und Mailinglisten bestehen. Daher basiert die Kommunikation, unabhängig vom Internetdienst, letztlich auf Text. Damit weicht die Ausgestaltung der sozialen Beziehungen in den Produktionsorten erheblich von denen im realen Raum ab. Zwar sind auch hier die Aspekte des aufeinander bezogenen Handelns und des Sinngehaltes gegeben (vgl. Weber 1978: 305), jedoch ist die für soziale Beziehungen wesentliche Kommunikation anders strukturiert, denn sie hat eine schmalere Bandbreite als bei face-to-face Begegnungen.

5.1.1 Kommunikationsmedien

In den folgenden Abschnitten sollen die verschiedenen verwendeten Kommunikationsmedien vorgestellt werden. Dabei soll auf deren Funktionsweise, die Art der Kommunikation und die Kontrollmöglichkeiten eingegangen werden. Peter Kollock und Marc A. Smith weisen darauf hin, dass diese Medien vielgestaltige Kommunikationsformen gestatten. Neben den bekannten Formen one-to-one und one-to-many ermöglichen sie teilweise auch many-to-many Interaktionen und können als Gruppenmedien bezeichnet werden (Kollock/Smith 1999: 3).

Die Untersuchung hat allerdings ergeben, dass es auch zu direkten Kontakten kommt. So gibt es innerhalb von einigen Projekten organisierte Treffen. Die Räumlichkeit beeinflusst natürlich auch die mögliche Ausgestaltung der sozialen Zusammenhänge.

5.1.1.1 Email und Mailinglisten

Email und Mailinglisten sind die älteste Form von Kommunikation im Internet. Während bei Email eine Nachricht von einer Person zu einer anderen gesendet wird, schickt bei Mailinglisten eine Person eine Nachricht an die Liste, von wo sie an alle Listenmitglieder weitergeleitet wird. Ein Thread (Diskussionsstrang) entsteht, wenn eine Person auf die Nachricht einer anderen antwortet und sich eine Diskussion entwickelt. Es werden in der Regel Nachrichtenarchive von Mailinglisten im WWW angelegt, so dass man auch später nachvollziehen kann, was jemand gesagt hat. Die

Kommunikation durch Mailinglisten ist asynchron, so dass eine Diskussion geführt werden kann, ohne dass die Beteiligten sich zu einer bestimmten Zeit treffen. Dies ermöglicht eine Kooperation über verschiedene Zeitzonen hinweg. Mailinglisten gehören normalerweise einer Person oder einer Gruppe, welche auch die Kontrollmacht über die Liste haben. Die Kontrolle wird an dem zentralen Punkt ausgeübt, an den alle Nachrichten geschickt, bzw. vom dem sie weitergeleitet werden. So kann darüber bestimmt werden, wer Nachrichten an die Liste schicken und wer die Liste abonnieren darf, und es können auch einzelne Nachrichten nicht weiter geleitet werden. Wie sehr diese Kontrollmöglichkeiten angewendet werden, ist sehr unterschiedlich (Kollock/Smith 1999: 5). Nur in den seltensten Fällen wird wohl der Moderator einer Liste alle Nachrichten durchsehen, bevor er sie weiterleitet.

5.1.1.2 Newsgroups

Newsgroups bilden Usenet, das ein eigener Internetdienst wie das www, oder auch Mail ist. Das Usenet setzt sich aus einer verteilten Nachrichtendatenbank zusammen. Anders als bei Mailinglisten werden den Empfängern die Nachrichten nicht einfach zugeschickt, sondern sie müssen sich in Usenet begeben und die einzelnen Newsgroups einsehen. Die Kommunikation durch Newsgroups ist wie bei Mailinglisten asynchron. Das Usenet hat keine zentrale Autorität, anders als bei Mailinglisten gehören Newsgroups auch niemanden.Allerdings haben Newsgroups auch ohne zentrale Autorität eine Ordnung und Struktur (Kollock/Smith 1999:5f.).

5.1.1.3 Internet Relay Chat

Der *Internet Relay Chat* (IRC) ist das größte nicht-kommerzielle Chatsystem. Bei Chats erfolgt eine synchrone Kommunikation. Eine Reihe von Personen befinden sich zur gleichen Zeit in dem gleichen Chat-Chanel und unterhalten sich miteinander. Dies geschieht, anders als bei Mailinglisten oder Newsgroups nicht durch das Verschicken ganzer Nachrichten, sondern die Tastatureingaben werden in Echtzeit übertragen. Die Kontrolle in Chat Channels ist sehr hoch: jeder Channel hat einen oder mehrere Operator, die die bestimmte Rechte (OP-Rechte) haben: sie können andere Personen aus dem Channel hinauswerfen (auch sich gegenseitig), bestimmen, wer den Channel betreten darf oder wie viele Personen sich im Channel aufhalten dürfen (Kollock/Smith 1999: 6).

5.1.1.4 Webseiten

Jedes Freie Software Projekt hat eine eigene Webseite. Dort lasen sich alle mögliche Informationen zu dem Projekt finden: Informationen zur entwickelten Software (Funktion, Updates, Dokumentation), aber auch Informationen über das Projekt

als solches, seine Geschichte, die Entwickler, die Projektstruktur, und wie man sich beteiligen kann usw.. Außerdem gibt es häufig Diskussionsforen, in denen sich Entwickler, aber auch Nutzer untereinander austauschen können. Zudem ist der Quellcode über diese Webseiten zugänglich (Ettrich 2004: 187). Webseiten dienen der Selbstauskunft von Projekten. Die Kommunikation mittels Webseiten ist unidirektional, der Inhalt wird von den Projektbeteiligten den Empfängern präsentiert. Die einzige Möglichkeit der bidirektionalen Kommunikation sind Diskussionsforen.

5.1.2 Andere Werkzeuge

Neben den Kommunikationsmedien gibt es noch weitere Werkzeuge, die die gemeinsame Arbeit der Entwickler erleichtern. Drei sehr wichtige und weit verbreitete werden im folgenden erläutert. Zudem soll noch ein Infrastruktur-Provider vorgestellt werden.

5.1.2.1 Source Code Management Systems (SCM)

Das SCM ist das zentrale Depot für den Programmcode, die Graphiken, die Übersetzungen und die Dokumentation eines Projekts, d. h. hier befindet sich die der Code und alles was zu dem Programm dazugehört. Das am häufigsten eingesetzte System ist das *Concurrent Versions System* (CVS).[1] SCMs ermöglichen mehreren Entwicklern gleichzeitig am Code zu arbeiten und die Änderungen der anderen nachzuvollziehen. Jeder kann verfolgen, welcher Code von wem zu welchen Zeitpunkt in das Programm eingefügt wurde (Ettrich 2004: 188). Es ist auch möglich, Änderungen wieder rückgängig zu machen. Mit Hilfe von SCMs können Entwickler direkt am Stamm-Quellcode arbeiten, sofern sie einen SCM-Account bzw. Schreibrechte haben. Außerdem ist es möglich, dass mehrere Entwickler zur selben Zeit unabhängig voneinander am selben Stück Code arbeiten. Welcher Code in so einem Falle dann letztlich im SCM übernommen wird, wird durch bestimmte Regeln innerhalb des SCMs bestimmt. Die Kommunikation in einem SCM besteht ausschließlich aus Code.[2] Anders als bei Email, IRC oder Webseiten strukturiert nicht ein Mensch den Inhalt, sondern ein Programm. Die Beteiligten müssen nicht einmal wahrnehmen, dass sie miteinander kommunizieren.

[1] http://www.cvshome.org

[2] Letztlich basiert jede Kommunikation im Internet auf Code, da alle Kommunikationsbeiträge (*Text*) in digitaler Form umgewandelt werden und habe Code sind. Hier ist aber nicht in Code umgewandelter Text gemeint, sondern Code im eigentlichen Sinne, als Quellcode von Software.

5.1.2.2 Bug Tracking Systems

Bug Tracking Systems sind Programme zur Verwaltung von Fehlermeldungen, die im Internet laufen. Das verbreitetetste ist wohl Bugzilla.[3] Neben Daten zu den Fehlern (Programm-Version, benutztes System und unter welchen Umständen sie auftreten) ist es möglich noch weitere Nachrichten bezüglich des Fehlers als frei formulierten Text zu schreiben. Alle Nachrichten zu einem bestimmten Fehler erscheinen dann am Ende der Seite zu diesem Fehler. So ist es möglich, wie in einer Art rudimentären Forum zu kommunizieren.

5.1.2.3 Mailinglistenarchive

Es gibt im Internet verschiedene Orte an denen alle Beiträge von Mailinglisten und Newsgroups archiviert werden. Diese sind für die Projektarbeit sehr wichtig, um auch nach Jahren nachvollziehen zu können, was schon mit welchem Ergebnis diskutiert wurde. Das ist besonders für Neue hilfreich, da sie die vorherige Entwicklung nicht selbst mitbekommen haben und so feststellen können, ob ihr Anliegen schon einmal durchdiskutiert wurde. Hierzu muss man sich das ganze Webarchive der Liste herunterladen und dann mit Volltextsuche nach signifikanten Schlüsselwörtern suchen. Solche Archive werden beispielsweise unter *Mailing list ARChives* (MARC)[4] für Mailinglisten oder Google Groups[5] für Newsgroups angelegt. Normalerweise steht auf der Homepage eines Projektes ein Hinweis (bzw. es befindet sich dort ein Link), wo die Projektmailinglisten archiviert werden.

5.1.2.4 SourceForge

Die Website SourceForge.net[6] bietet Projekten die Möglichkeit sich dort anzusiedeln. Diese Möglichkeit wird gut angenommen, so lassen sich zur Zeit über 88.000 Projekte dort finden.[7] Besonders kleinere Projekte nutzen die Chance einen kostenlosen Host zu haben.[8] Außerdem bietet SourceForge ihnen die technische Infrastruktur, wie Mailinglisten, SCMs, Foren etc. SourceForge gehört zur Open Source Technology Group, die neben SourceForge weitere Serviceleistungen für Projekte, Entwickler und Nutzer Freier Software bereitstellt.

[3] http://www.bugzilla.org
[4] http://marc.theaimsgroup.com
[5] http://www.google.de/grphp?hl=de&tab=wg&q=
[6] http://sourceforge.net/index.php
[7] http://sourceforge.net/index.php am 2.10.2004.
[8] Hosten bedeutet, jemanden Platz auf einem Internetserver zur Verfügung zustellen, um dort die Daten für die Webseite, Chat-Channels etc. zu speichern. Neben dem Platz für die Daten braucht ein Projekt noch genug Serverkapazitäten, um einen gegebenenfalls starken Zugriff (Traffic) bewältigen zu können.

5.2 Die Projekte

Die Produktionsorte sind meist aus Freiwilligen zusammengeschlossene Projekte. Die Projekte entstehen meistens dadurch, das jemand eine Problemlösung sucht oder einfach herumexperimentiert und das Interesse von anderen anzieht, die sich dem Vorhaben anschließen.

Die Projekte werden aber nicht nur von Privatpersonen gegründet, so gibt es mittlerweile auch einige (wenige) Unternehmen, die den Quellcode ihre Software freigeben. Bekannte Beispiele sind hier Mozilla[9] (Netscape) oder auch OpenOffice[10] (SUN).[11] Es kommt auch vor, das Unternehmen Projekte materiell oder durch Mitarbeiter unterstützen, weil sie eine bestimmte Software für hilfreich halten, diese aber nicht selbst als Teil ihres Produktes herstellen wollen. Die Größe der Projekte variiert stark, es beginnt mit Ein-Personen-Projekten bis zu solchen, die über tausend Personen umfassen. Die thematische Spannweite der Projekte ist sehr groß, es gibt ganze Betriebssysteme, einzelne Kernels, Desktopsysteme, Applikationen (Anwenderprogramme) aber auch Spiele und einzelne Gerätetreiber.

Nach Eric Raymond gibt es drei Wege Eigentümer eines Freier Software Projekts zu werden:[12] ein Projekt selbst zu gründen, das Projekt von dem vorherigen Eigentümer übergeben zu bekommen oder sich ein eigentümerloses Projekt anzueignen. Letztes geschehe folgendermaßen: zu erst versucht man den vorherigen Eigentümer ausfindig zu machen. Möglichst sollte dies über verschiedene Foren passieren, zum Beispiel Newsgroups oder auch Mailinglisten. Wenn sich jemand in dieser Zeit als Eigentümer meldet, hat der Aspirant das Nachsehen (Raymond 2000b: 7). Es ist aber aufgrund der Lizenzen immer möglich ein paralleles Projekt zu gründen, falls es Unstimmigkeiten mit dem Projektleiter bzw. -eigentümer gibt.

Im Folgenden sollen fünf Projekte vorgestellt werden, um einen Einblick in die Heterogenität des Feldes zu geben.

5.2.1 Fallbeispiele

5.2.1.1 Linux

Das Linux-Projekt wurde 1991 von Linus Torvalds ins Leben gerufen. Seine ersten Mitarbeiter bekam er über die Minix Newsgroup. Das Projekt erstellt den Kernel

[9] http://www.mozilla.org

[10] http://www.openoffice.org

[11] Allerdings werden solche Fälle hier von der Betrachtung ausgenommen, da dies den Rahmen der Arbeit in nicht zu bewältigender Weise erweitert hätte. Zur Geschichte von Mozilla siehe Moody 2001: 269ff.; Raymond 2000a.

[12] Der Eigentümer eines Projekts ist nicht mit den Eigentümern des Codes zu verwechseln. Das Eigentum eines Projekts bezieht sich auf das Projekt (die Produktionsumgebung) und nicht auf das Produkt.

des Betriebssystems und ist damit eines der wichtigsten und zentralsten Projekte. Aufgrund seiner Wichtigkeit und des immer größeren Funktionsumfangs hat das Projekt heute schätztungsweise mehr als 1000 Mitarbeiter und ist eines der größten überhaupt. Während zu Beginn des Projekts fast täglich neue (rudimentäre) Versionen veröffentlicht wurden, gibt es heute etwa im Jahresrhythmus eine neue stabile „große“ Version (die an den geraden Versionsnummern erkennbar sind – *2.2*, *2.4* usw). Die Frequenz der Veröffentlichung von Versuchsversionen für Entwickler ist deutlich höher.

Die personelle Organisation des Projekts hat die Gestalt einer hierarchischen Baumstruktur:[13] Für jede große Version gibt es einen so genannten Kernel-Maintainer (beim 2.6er Kernel war das Linus Torvalds selbst, bei der Version 2.4 war es Marcelo Tosatti), bei dem alle Fäden zusammenlaufen. Als nächste Ebene gibt es Maintainer für verschiedene funktionale Teilbereiche, zum Beispiel Netzwerke. Innerhalb dieser Bereiche gibt es eine weitere funktionale Unterteilung und wiederum Maintainer, die diese Bereiche leiten. Auf dieser Ebene gibt es mehrere gut frequentierte offene Mailinglisten, an die auch neue und außenstehende Entwickler ihre Beiträge schicken können. Hier wird sehr viel in Form konkreten Programmcodes kommuniziert, allgemeine abstrakte Diskussionen spielen eine kleinere Rolle. Es gibt eine hohe Erwartung, dass Beitragende konkrete Implementierungen liefern – das Motto heißt „show me the code!“. Die meisten Entwicklungen kommen aus einem eher kleinen Kreis, auch wenn es immer wieder einzelne Problemlösungen und Korrekturen von anderen Personen gibt. Wenn jemand ein Stück Code in den Kernel hineinbekommen möchte, muss er es dem Maintainer der jeweiligen untersten Ebene schicken, der es prüft und eventuell in seine Version einbaut. Diese neue Version schickt er an den nächst höheren Maintainer weiter, der es wiederum bei sich integriert bis es schließlich den Maintainer des Kernels erreicht.

Die Autorität der Urteile der einzelnen Maintainer ist sehr hoch. Sie alleine entscheiden, was sie übernehmen und was nicht. Theoretisch ist es möglich, bei Kontroversen mit dem Maintainer eine Diskussion auf einer Mailingliste anzufangen oder direkt den nächst höheren Maintainer zu kontaktieren. Nach Wissen meines Informanten ist allerdings noch niemand wegen Unstimmigkeiten übersprungen worden. Allerdings würde auch hin und wieder eine Abkürzung in der Baumstruktur entstehen, wenn ein Maintainer zu beschäftigt ist oder nicht reagiert. Letztlich wollen die Maintainer aber nur sehr wenige Ansprechpartner haben, da es sonst zu unkoordiniert zu gehe und niemand mehr den Überblick habe. Neben dieser hierarchischen inhaltlichen Qualitätskontrolle kommen verschiedene automatisierte Systeme zur Korrektur von trivialen Fehlern (z. B. Tippfehler) zum Einsatz, die über Email die Entdecker dieser Fehler über die Übernahme ihrer Meldung informieren.

[13] Die folgenden Informationen stammen aus einem Interview mit einem Mitarbeiter von Linux, vom 25.6.2004 auf dem Linuxtag in Karlsruhe.

Eine Möglichkeit über Kernel-spezifische Themen zu kommunizieren ist die unmoderierte Hauptmailingliste des Projekts (linux-kernel@vger.kernel.org). Auffällig ist, dass es keine klassische Projekt-Webseite gibt. Auf der Seite http://www.kernel.org/ befinden sich nur der Quellcode, Links zur Installationshilfen, sonstigen Informationen über Linux und einer *Frequently Asked Questions* (FAQ) Seite für die Kernel-Mailingliste. Es gibt aber keinerlei Informationen über das Projekts mit Ausnahme des Quellcodes.[14] Auf der FAQ Seite lässt sich auch die Begründung für die untypische Webseite finden, es gäbe kein richtiges Linux Kernel-Team , sondern eher eine sehr große Zahl von Beitragenden, so dass eine typischen Seite kaum umzusetzen sei. Die Begründung endet mit den Worten:“ Finally, altough this is not a rule, most Linux kernel contributors prefer to keep a low profile, for various reasons.“ (Gooch et al. 2004).

5.2.1.2 Debian

Debian,[15] eine Linux Distribution, wurde 1993 von Ian Murdock ins Leben gerufen, weil er von der damals verbreitetetsten Linux-Distribution *Softlanding Linux System* (SLS) von Peter MacDonald enttäuscht war (Murdock 2003). Um die Probleme von MacDonald, der alles alleine machte und nicht mehr mit er Arbeit hinterher kam, zu vermeiden, lud er Interessierte zur Mitarbeit ein. Debian besteht aus einer Vielzahl von einzelnen Paketen,[16] für die einzelne Personen, die Maintainer die Verantwortung übernehmen. Damit die Pakete sich in das gesamte System integrieren lassen, sind Standards und Regeln für sie definiert worden (Moody 2001: 124f.). Als eine Distribution ist Debian als Netzwerk eher flach und in die Breite angelegt. Für jedes Programm, das in Debian integriert wird, gibt es mindestens einen Maintainer. Der Begriff ist bei Debian also weniger die Bezeichnung einer höher geordneten Position, als der Hinweis, dass jemand für ein Paket zuständig ist. Es gibt um die 1000 Mitarbeiter bei Debian.[17] Die Kommunikation verläuft vor allem über Mailinglisten, zuweilen verabreden sich verschiedene Unterprojekte auch zu bestimmten Zeiten in IRC-Channels. Besonders umfassend ist die Webseite von Debian. Hier lassen sich eigentlich alle relevanten Informationen über das Projekt finden. Hintergrund dieser Öffentlichkeit ist die Einstellung, dass ein freies Projekt nicht nur den Code veröffentlichen soll.

[14] In einer Datei im Quellcodeverzeichnis befindet sich eine Liste mit den Namen der Maintainer. Um diese Liste zu finden muss man sich den gesamten Kernel herunterladen und nach dieser Datei gezielt suchen, d.h. auch hier ist die Information nicht gut zugänglich.

[15] http://www.debian.org. Der Name setzt ich aus den Vornamen von Ian Murdock und seiner Frau Deb zusammen.

[16] Fertige Linux-Programme werden häufig in Form von Paketen verteilt. Diese enthalten eine lauffähige Version des Programmes selbst und meist auch notwendige Bibliotheken. Wenn der Benutzer ein Paket installiert, wird ihm zudem gemeldet, wenn sein System andere, notwendige Programme nicht zur Verfügung stellt.

[17] Vortrag von Martin „Joey“ Schulze „(Software-)Entwicklung im Debian-Projekt auf dem Linuxtag in Karlsruhe am 12.7.2003.

Heute hat Debian wahrscheinlich die ausdifferenzierteste und am stärksten kodifizierte Organisationsstruktur im Bereich Freie Software. Es gibt zwei verfassungsartige Schriften: *Debian Social Contract*[18] und *Debian Constitution*[19]. Der Social Contract zeigt die Beziehung zwischen den Debian Mitarbeitern und der „Entwicklern von Freier Software" auf und welche lizenztechnischen Voraussetzungen eine Software erfüllen muss, damit sie ins Debian System integriert werden kann. Die Debian Constitution regelt, wie bestimmte Positionen besetzt werden und wer welche Aufgaben und Kompetenzen hat.

Innerhalb der Organisation von Debian gibt es drei Gruppen von Entwicklern: wenige mit offiziellen Ämtern, die *Debian Developer* (was ein regelrechter Titel ist) und freie Beitragende. An der Spitze steht der Projektleiter, der so genannte *Debian Project Leader* (DPL), dieser repräsentiert das Projekt nach außen und ist für die innere Koordination und Kommunikation verantwortlich. Der DPL wird von den Debian Developers aus ihrer Mitte für den Zeitraum eines Jahres gewählt. Neben dem DPL gibt es eine Reihe weiterer offizieller Funktionen, beispielsweise den Project Secretary, der von seinem Vorgänger und vom DPL ernannt wird.[20] Der Projekt Secretary ist für alle Abstimmungen, über Personen und auch Themen, und für die Interpretation der Verfassung verantwortlich. Außerdem gibt es noch *The Technical Committee*. Die Mitglieder werden entweder von den bereits im Amt befindlichen Mitgliedern oder vom DPL bestimmt. Es kümmert sich um technische Sachfragen und legt die grobe Entwicklungsrichtung fest. Außerdem kann es einem Entwickler eine bestimmte Entwicklung untersagen. Neben diesen offiziellen Funktionen gibt es noch eine Reihe von Teams, die sich mit bestimmten Aufgaben beschäftigen, z. B. mit der Projektinfrastruktur (FTP Master, List Master) oder auch auch dem Qualitätssicherung oder dem Release Management (Michlmayr 2003).

Um ein offizieller Debian Developer zu werde, muss man eine Prüfung bestehen, die aus mehreren einzelnen Tests besteht. Erst nachdem man diese Prüfungen bestanden hat, ein Prozess der einem Monat oder auch über ein Jahr dauern kann, ist man offizieller Debian Entwickler (Debian 2003). Erst dann bekommt man einen CVS-Account, d.h. man kann selbst seinen Code in das Projektverzeichnis einchecken, man kann Sponsor für andere Entwickler ohne CVS Account werden und deren Code für sie ins System integrieren, man kann an den Wahlen teilnehmen und eine offizielle Funktion bekleiden. Diese Aufnahme ist – meines Kenntnisstandes nach – recht einzigartig (vgl. Kap. 6.2.2).

Schließlich gibt es noch viele Entwickler, die zwar bei Debian mitarbeiten, aber keine offiziellen Entwickler sind. Wenn sie ihren Code in das System integrieren möchten,

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

[19] http://www.debian.org/devel/constitution

[20] Für den Fall von Unstimmigkeiten ist ebenfalls eine Prozedur festgeschrieben, dann bestimmt der Vorstand der *Software in the Public Interest* (SPI), die Organisation von Debian für Rechtsfragen, ernannt.

müssen sie sich einen Sponsor suchen, der das in seinem Namen für sie macht. Zum Teil gibt es auch regelrechte kleine Gruppen die sich gemeinsam um größere Pakete kümmern.

5.2.1.3 KDE

KDE ist ein Desktopsystem für Unix-Systeme, die den Gebrauch für unbedarftere Nutzer stark vereinfachen. Nach Moody entstand KDE aus dem Versuch von Mattias Ettrich seiner Freundin Linux näher zu bringen. Da KDE auf der (damals) nichtfreien Bibliothek Qt von Trolltech aufbaut,[21] war es schnell Gegenstand einer größeren Auseinandersetzung unter den Entwicklern Freier Software, aus der auch das „freie" Konkurrenzprojekt *GNU Network Object Model Environment* (Gnome) hervorging (Moody 2001: 358ff). Heute gehört KDE mit über 1000 Beteiligten, zu den größten Projekten im Bereich Freier Software (Brand o. J.: 6). Kommuniziert wird hauptsächlich über Mailinglisten, aber auch über Chat (ebd.: 41ff.). Zusätzlich gibt es vor den Releases kleinere Entwicklertreffen und es werden projektunabhängige Treffen, wie Linux-Konferenzen oder Messen zum gemeinsamen Programmieren genutzt (Brand ebd.: 8, 43f.).[22] Das Projekt hat eine sehr ausdifferenzierte Webseite, die sich sehr auf die Nutzer konzentriert. Daneben gibt es aber auch einen Bereich für die „Community", in dem beispielsweise projektinterne Nachrichten oder auch Interviews mit einzelnen Entwicklern gestellt werden. Über die Projektorganisation lässt sich allerdings kaum etwas finden.

Das Projekt teilt sich in mehrere Unterprojekte auf, die selbst weiter aufgegliedert sind. KDE umfasst neben dem grundlegenden System auch eine Vielzahl von Anwendungsprogrammen die darauf aufbauen. Wichtige Bereiche sind die Haupt- und Nebenbibliotheken und die einzelnen Programme. Daneben gibt es noch die Bereiche der Übersetzer und der Dokumentierer (ebd.: 11). Jedem Unterprojekt steht ein Maintainer vor, der meistens der Gründer des Unterprojekts ist. Die Organisation orientiert sich an den aufkommenden Aufgaben, und daraus ergeben sich auch Positionen (ebd.: 32). Wenn ein Maintainer seine Arbeit einstellt, soll er sich selbst einen Nachfolger suchen. Wenn ihm niemand geeignetes bekannt ist, sind ihm dabei langjährige und sehr stark eingebundene Entwickler behilflich (ebd.: 34f.). Falls niemand gefunden wird, verwaist das Projekt und es fällt irgendwann aus dem System heraus – wichtige Unterprojekte finden eigentlich immer einen neuen Maintainer, da sie im Zweifelsfall jemand aus Pflichtgefühl übernimmt. Innerhalb der Projektes gibt es einen informellen inneren Kreis, dem langjährige und verdiente Entwickler

[21] Mittlerweile hat Trolltech Qt zusätzlich unter die GPL und die *Q Public License* (QPL) gestellt (Trolltech o. J).

[22] Auf den Messen haben die Projekte, nicht nur KDE, meist eine Ecke mit einigen Rechnern, die zur Demonstration von Programmen gedacht sind, die aber auch zum gemeinsamen Arbeiten genutzt werden.

angehören. Sie treffen auch die strategischen Entscheidungen für das gesamte Projekte sie. Außerdem rekrutieren aus ihrer Mitte den Releasemaintainer.[23] Um die finanzielle und rechtlichen Belange des Projekts kümmert sich der 1997 gegründete Verein KDE e.V. (KDE o. J.).

Der Einstieg in das Projekt ist einfacher als bei Debian. Interessierte müssen zunächst einmal ihren Willen zur Mitarbeit durch das Beisteuern von zwei bis drei Patches belegen. Wenn diese vom Verantwortlichen für gut befunden werden, kann er für den Interessierten einen SCM-Account beantragen (ebd.: 23f.). In das Projekt ist er integriert, wenn er eine Projekt-Emailadresse erhalten hat. Um in den inneren Kreis zu gelangen, muss ein Entwickler über Jahre kontinuierliche hochqualitative Arbeit leisten. Es gibt keine formale Aufnahme, d.h. irgendwann ist es einfach geschehen.

5.2.1.4 Eisfair

Eisfair ist ein Internetserver für DSL-Verbindungen, der auf Linux läuft. Das Projekt entstand 2001 aus dem Schwester-Projekt *Floppy ISDN for Linux* (fli4l),[24] das ebenfalls von Frank Meyer initiiert wurde. Da fli4l immer weiter ausgebaut wurde und schließlich mehr Funktionen übernahm als geplant war, wurde der Bereich des Internetservers nach Eisfair ausgegliedert (Meyer 2002).

Eisfair ist ein eher kleines Projekt, es sind alles in allem vielleicht 100 Personen.[25] Der aktive Kern besteht aus 10-20 Entwicklern. Der Gründer, Frank Meyer, ist der Projektleiter. Die anderen Entwickler sind jeweils für einen eigenen Bereich verantwortlich und arbeiten größtenteils autonom. Punkte, die mehrere Entwickler betreffen werden gemeinsam durchgesprochen, wobei der Projektleiter allerdings das letzte Wort hat. Meistens werden Entscheidungen aber dergestalt getroffen, dass nach Einwänden gefragt wird und die Anliegen ausdiskutiert werden. Der Zeitraum bis zu einer Entscheidung beträgt zwei bis drei Wochen, falls jemand mal für ein paar Tage nicht erreichbar ist. Die Kooperation findet hauptsächlich über Newsgroups statt. Für wichtige Diskussionen treffen sich die Entwickler zu einer verabredeten Zeit im zum Projekt gehörigen IRC-Channel.

[23] Der Releasemaintainer hat eine zentrale, sehr arbeitsintensive Funktion inne. Er kümmert sich um die gesamte Koordination und den Ablauf für Releases. Da die Arbeit das Ausmaß eines Ganztagsjob hat, werden die Releasemaintainer häufiger von Unternehmen (z. B. kommerziellen Distributoren) bezahlt, die an KDE ein geschäftliches Interesse haben. Der Releasemaintainer rekrutiert sich aus dem inneren Kreis und ist eine Person, die sich selbst dafür meldet – aufgrund des großen Arbeitsaufwandes hat der Kandidat in der Regel keine Konkurrenz bzw. alle sind froh, wenn jemand die Aufgabe übernimmt. Um den Releasemaintainer schart sich eine Gruppe von Entwicklern, die ihn bei seiner Aufgabe unterstützt (Brand o. J.: 38ff.).

[24] fli4l ist ein einfacher Router ursprünglich für ISDN. Mittlerweile ist er auch für DSL und Ethernet einsetzbar (fli4l o. J.).

[25] Die folgenden Informationen über Eisfair stammen aus einem Interview mit einem Mitarbeiter von Eisfair in Karlsruhe am 13.7.2003.

Besonders an Eisfair ist, dass das Projekt auf Deutsch arbeitet. Zu Beginn war das Projekt englischsprachig, aber es haben sich nur Deutsche gemeldet, so dass schon bald die Sprache gewechselt wurde.[26]

5.2.1.5 WorldForge

WorldForge[27] ist ein Projekt zur Erstellung von Multi-Player Rollenspielen. Es ist 1998 aus dem Projekt Altima hervorgegangen, das um Avinash Gupta entstand (Farlex 2004). Neben – noch nicht fertig gestellten – Spielen entwickelt WorldForge Game-Engines,[28] Grafiken, Musik, Regelwerke und ähnliches, aus dem sich Interessierte selbst ihre eigenen Spiele zusammenbauen können. Der Kern des Projekt besteht aus ungefähr 15 Personen, während die „Community“ etwa 50 umfasst.[29]

Das Projekt ist in verschiedene Teilbereiche unterteilt: Programmierung, die Spielwelten und Infrastruktur. Zur Programmierung gehört alles Technische im Hintergrund von Spielen (Server, Clients, Bibliotheken), zu den Welten was Spieler sehen (Graphiken, Regelwerke, Musik) und zu Infrastruktur, alles was das Projekt am Laufen hält (CVS, IRC-Channels, Webseite, Mailinglisten).

WorldForge hebt sich nicht nur durch sein Thema (Spiele) von anderen Projekten ab, sondern auch durch seine Struktur. Es gibt keinen Projektleiter – der letzte fuhr in den Urlaub, übergab seine Aufgabe einem anderen und kehrte nie wieder zurück. So ist die Situation bis heute. Da der „Stellvertreter“ nicht wirklich die Rolle eines Leiters einnahm, regelt das Projekt alles informell. Es scheint allerdings einen engeren Kreis von Entwicklern zu geben, der mehr Einfluss als andere hat.[30] Die einzelnen technischen Unterprojekten haben allerdings noch Maintainer. Ungewöhnlich ist auch, dass nicht nur Entwickler und Nutzer als zum Projekt als gehörig betrachtet werden, sondern auch Sympathisanten, die sich einfach gerne im dazugehörigen IRC-Chanel aufhalten und die Idee des Projekts gut finden. Aussagekräftig ist auch die Tatsache, dass das wichtigste Kommunikationsmedium IRC ist.

[26] Eisfair ist das einzige Projekt, das mir bekannt ist, dass nicht Englisch als Projektsprache hat.

[27] http://worldforge.org

[28] Moderne Computerspiele bestehen heute meist aus einer Serversoftware (oder engine), die die grundlegende Logik des Spiels und zahlreiche Funktionen (Bewegungen der Spielfiguren, Regeln, etc.) beinhaltet, und einer darauf aufbauenden Komponente, die das Konzept in ein konkretes Spiel umsetzt. Diese Architektur ermöglicht auch leicht Multiuser-Spiele im Internet.

[29] Die folgenden Informationen stammen aus einem Interview mit einem Projektmitarbeiter vom 12.7.2003 vom Linuxtag in Karlsruhe.

[30] Mein Informant berichtete mir über einen Beschluss über den Ausschluss einer Person. Bei dieser Diskussion schienen nur die wichtigsten Beteiligten involviert gewesen zu sein, daraus schließe ich, dass es informelle hierarchische Unterschiede zwischen den Beteiligten gibt.

5.2.2 Organisations- und Entscheidungsstrukturen

Die Projekte sind also recht unterschiedlich. Neben der Größe und dem thematischen Feld ist ein wichtiges Unterscheidungskriterium die innere Organisation. Zu ihrem Verständnis sind Grad der Formalisierung und Ausgestaltung der Entscheidungs- bzw. Machtstrukturen, die beide miteinander in Bezug stehen, wesentliche Kriterien. Zur Analyse erscheint es sinnvoll, die beschriebenen Projekte als politische Systeme aufzufassen. Dabei ergeben sich drei Typen – Demokratie, Häuptlingstum und akephale Ordnung. Es soll auch auf die Struktur weiterer Projekte eingegangen werden, die noch weitere Typen ergeben.

Die Ämterstruktur samt ihrer Aufgaben und Kompetenzen sowie die Verfahrensweisen von Debian sind niedergeschrieben. Der Projektleiter wird von den Mitgliedern gewählt und über wichtige Entscheidungen wird abgestimmt. Beides folgt einem formalen Verfahren. Andere Offizielle, wie der Project Secretary und die Mitglieder des Technical Committees, werden ernannt. Man kann hier von einer repräsentativen Demokratie sprechen. Dazu gehört auch, dass nur anerkannte Mitglieder des Projektes (die Debian Developer) das Wahlrecht haben und somit gewissermaßen als „Bürger“ gegen sonstige Personen abgegrenzt sind. Das Vorhandensein einer Verfassung unterstreicht dies schon in der Terminologie. Die Transparenz aller das Projekt betreffenden Vorgänge ist ein weiteres Indiz für eine demokratische Form. Meines Wissens nach ist der Formalisierungsgrad von Debian recht einzigartig.

Linux, KDE und Eisfair haben zwar konsolidierte Strukturen, diese sind aber im Vergleich zu Debian in einem viel geringeren Maße differenziert und kodifiziert. Die Funktionsträger sind anders als bei Debian nicht formal gewählt, sondern beziehen ihre Legitimation aus anderen Quellen. Hauptsächlich sind dies Tradition oder Charisma, wenn man Webers Terminologie folgt (vgl. hier und im folgenden Weber 1978: 222ff.; Weber 1980: 130ff.). Die leitenden Positionen werden hier entweder auf traditionale Weise besetzt, indem sie von den Gründern eingenommen werden, oder, bei der Findung neuer Leiter, durch charismatische Kriterien. Das Charisma beruht vor allem auf der „Gnadengabe“ der großen Fähigkeiten als Entwickler, also auf der Reputation, ein guter Programmierer zu sein. Auch die Reputation, über ausgeprägte Koordinations- und Teamfähigkeit zu verfügen trägt dazu bei. Diese Legitimation erhalten sie durch die Zustimmung der anderen Entwickler. Sie muss ständig erhalten bzw. bestätigt werden, sonst verlieren die Leitungspersonen ihre Reputation und damit ihren Einfluss (bezüglich KDE vgl. Brandt o. J.: 27f). Bei Linux und bei Eisfair sind starke traditionale Veranlagungen erkennbar. Bei KDE funktioniert die personelle Strukturierung eher über Charisma.

Projekte dieses Types erinnern an Häuptlingstümer oder Big Man Systeme. Big Men fungieren als Sprecher von egalitär strukturierten Gruppen. Sie qualifizieren sich aufgrund ihrer Persönlichkeit, d.h. wegen ihrer Energie, Tapferkeit, Leistungsfähigkeit,

ihrer rhetorischen Fähigkeiten und vor allem ihr Organisationstalent für diese Position. Das Amt des Big Man ist nicht erblich und muss – auch von einer bereits in der Funktion befindlichen Person – ständig neu erworben bzw. bestätigt werden. Er hat keinerlei Erzwingungsstab. Der Häuptling ist dagegen ein permanentes, festumrissenes und erbliches Amt.[31] Die Legitimation der Sonderstellung des Häuptlings lässt sich häufig auf mythologische Ursprünge zurückführen. Allerdings kann ein Häuptling auch durch die Ältesten abgesetzt werden. Häuptlingstümer sind hierarchisch geordnet. Neben Häuptlingen als Vorstehende einzelner Gruppen gibt es auch den Oberhäuptling (Paramount Chief), der unter sich weitere lokale Unterhäuptlinge hat, die sich um die konkrete Administration kümmern (Kohl 1993: 58ff). Eisfair, gehört zwar zu dieser Kategorie, allerdings ist es aufgrund seiner personellen Größe viel übersichtlicher strukturiert.

WorldForge repräsentiert hier einen Projekttyp mit nur rudimentären Strukturen. Es gibt keinen offizielle Gesamtleitung, sondern nur Maintainer von Unterbereichen. Damit könnte dieses Projekt als akephale Gesellschaft begriffen werden, in der es zwar Personen gibt, die mehr Einfluss als andere haben, aber alle Entscheidungen in direkter Absprache getroffen werden (ebd.: 53f). Bezeichnenderweise kooperieren die Mitarbeiter von Worldforge auch nicht über Mailinglisten, sondern über IRC, das der für akephale Gesellschaften typischen Versammlungsdemokratie ähnelt.

Neben den oben vorgestellten Projekten gibt es noch mindestens zwei weitere Typen: Ein-Personen Projekte, die naturgemäß keine innere Organisation haben. Sie finden sich meist auf der persönlichen Webseite der Entwickler und haben bestenfalls eine Mailingliste für die Nutzer. Projekte des anderen Typs werden von einem Coreteam geleitet. Dieses kann vom den Entwicklern des Projekts gewählt werden, wie es bei NetBSD der Fall ist.[32] Oder, wie es bei Netfilter praktiziert wird,[33] die bereits im Coreteam befindlichen können weitere Entwickler einladen. Der wesentliche Unterschied zu den oben beschriebenen Projekttypen ist, dass es keinen Projektleiter gibt, sondern eine kleine Gruppe bestehend aus vier oder oder Personen. Die sonstige Struktur kann einem der genannten Modelle folgen.

Angeregt durch Eric Raymonds „Homesteading the Noosphere“ aus dem Jahre 1998 wird häufig geschrieben, dass die personalpolitische Strukturierung mit Hilfe von auf meritokratischen Mechanismen basierender Reputationszuweisung geschieht (vgl. Raymond 2000b).[34] Die Erlangung von Reputation verläuft aber nicht nur über meritokratische Mechanismen, sondern auch über Elder-Vorstellungen (ebd.: 7). Es stellt sich hier die Frage, wie weit es zu Fraktionenbildung innerhalb von Projekten

[31] Natürlich ist Erblichkeit nur bedingt auf Freie Software übertragbar, dennoch könnte man als solche die Projektleitung durch den Gründer sehen, oder wenn dieser seinen Nachfolger bestimmt.

[32] Interview mit einem NetBSD Mitarbeiter am 5.4.2003 in Magdeburg.

[33] Interview mit einem Linux Mitarbeiter am 25.6.2004 in Karlsruhe.

[34] Diese Beobachtung konnte ich seit 1999, dem Beginn meiner Beschäftigung mit dem Thema, machen. Zunächst gab es eigentlich nur Raymonds Essays, nach einer Weile erschienen weitere Texte von verschiedenen Personen, die letztlich Raymonds Thesen nur wiederholten.

kommt, die versuchen, Positionen mit Personen, die ihnen nahe stehen, zu besetzen, und nicht nur aufgrund von deren Leistungen.[35] Die häufige Gleichsetzung von Reputation und Meritokratie im Feld der Freien Software Entwickler ist also nur bedingt zutreffend. Vielmehr müssen verschiedene Bezugsgrößen für Reputation betrachtet werden (vgl. Wolfinger 1960: 638f.).

Abschließend ist anzumerken, dass die organisatorischen und politischen Strukturen der Projekte nicht wie bei Unternehmen am Reißbrett entworfen werden, sondern das Resultat von Zufall, Konventionen („bei Freien Software Projekten ist das halt so“) und Aushandlung sind. Sie werden weder aus prinzipiellen politischen Überlegungen hergeleitet, noch wird versucht, vorab Probleme und Aufgaben einzuschätzen und eine entsprechende Struktur geplant. Die Organisationsstrukturen sind stark von dem jeweiligen Aufgabenfeld und der Größe der Projekte beeinflusst und werden deren Veränderung spontan angepasst, d.h. es scheint ein Forms-follows-Function Prinzip vorzuherrschen.

5.3 Die Vernetzung des Feldes

Die Projekte stehen nicht vereinzelt da, sondern kooperieren mehr oder weniger stark miteinander. Dies trifft natürlich insbesondere auf Distributionen zu, die den Code von anderen zusammenstellen. Debian hat daher ständig Kontakt zu Entwicklern anderer Projekte, um technische und lizenzrechtliche Fragen zu klären. Aber auch bei anderen größeren Projekten gibt es einen permanenten Abstimmungsbedarf. So sind die Entwickler des Linux Kernels und diejenigen einzelner Module aufeinander angewiesen und stehen in dauernder Kommunikation, um ihre Programme aneinander anzupassen. Dies kann dazu führen, dass wichtige Entwickler eines Kernel-Moduls auch die entsprechende formale Position in der Hierarchie des Linux-Projektes übernehmen.

Neben den technischen Verbindungen gibt es auch Entwickler, die in verschiedenen Projekten gleichzeitig tätig sind. Es ist allerdings unklar, ob sie diese individuelle Doppelmitgliedschaft zur aktiven Förderung des Austauschs und der Zusammenarbeit zwischen den Projekten nutzen. Einige Projekte pflegen soziale Kontakte untereinander, so luden die Entwickler von KDE die Gnome und Debian Mitarbeiter zu ihrem Social Event während des Linuxtages 2004 in Karlsruhe ein. Es ist allerdings sehr fraglich, ob das ganze Feld durchgängig untereinander vernetzt ist.

Eine besondere Stellung innerhalb des Feldes haben Organisationen mit politischer Ausrichtung, wie etwa die Free Software Foundation (FSF) und ihre Schwesteror-

[35] Bisher gibt es keine harten Daten zu diesem Punkt, aber es wurde innerhalb von Gnome genau dieser Vorwurf gemacht. Ob der Vorwurf allerdings berechtigt ist, muss noch weiter geprüft werden. Vgl. insbesondere http://mail.gnome.org/archives/gnome-de/2004-April/msg00022.html; http://mail.gnome.org/archives/gnome-de/2004-April/msg00030.html.

ganisation FSF Europe oder die Open Source Initiative (OSI).[36] Diese Gruppen kooperieren in vielfältiger Weise, teilweise in formalisierten Assoziationen. Sie widmen sich hauptsächlich der Lobbyarbeit und der rechtlichen Vertretung. Projekte, in denen rechtliche Fragen auftauchen, wenden sich damit an die FSF. Ihre politische Arbeit geht dabei auch über den Bereich Freie Software hinaus, indem sie mit anderen Organisationen etwa Kampagnen gegen Softwarepatente planen. Kooperationspartner in diesem Bereich sind Vereinigungen wie die Electronic Frontier Foundation (EFF – Hauptthema: Redefreiheit im Internet) oder die Foundation for a Free Information Infrastructure (FFII – gegen Softwarepatente).

Aufgrund ihrer nicht-technischen Aufgabe finden diese Organisationen unter den Entwicklern zwar Anerkennung, werden aber als separate Ebene wahrgenommen. In ihnen beteiligen sich auch keineswegs nur Entwickler, Rechtswissenschaftler etwa sind dort auch oft anzutreffen. Das entspricht auch dem Selbstbild dieser Organisationen. So sieht es die FSF Europe als ihre Aufgabe an, den Raum für Freie Software zu schaffen, zu sichern und zu erweitern – ihn auszufüllen wird dann den Entwicklungsprojekten überlassen.[37] Allerdings bieten einige der politischen Organisationen verschiedenen Projekten ihre Infrastruktur an, so sind diverse Latex-Projekte etwa auf den Servern der FFII gehostet.

Die Projekte sind wie oben erwähnt die Produktionsorte, die Politik scheint aus ihnen weitgehend ausgelagert zu sein. Bei Debian findet sie zwar auch innerhalb des Projektes statt, aber nur auf einer eigenen Mailingliste (debian-legal@lists.debian.org), die von den technischen Listen getrennt ist. Auch haben inzwischen viele Projekte eigenen Vereine und Stiftungen gegründet, die sich aber auf die Verwaltung der jeweiligen Rechte und Finanzangelegenheiten beschränken. Weitergehende politische Aktivitäten werden dann aber den darauf spezialisierten Organisationen überlassen.

Schließlich existieren Kommunikationsforen, die über einzelne Projekte hinausgreifen und im Feld eine Art öffentliche Sphäre bilden. Dazu gehören einerseits Zeitschriften wie etwa das Linux Magazin oder online Medien wie Slashdot[38] oder Newsforge[39]. Auch die Linux User Groups gehören zu diesem Bereich. Alle diese Foren bieten die Möglichkeit, allgemeine Themen im Bereich Freier Software zu diskutieren.

[36] Gerade FSF und OSI arbeiten aufgrund ihrer unterschiedlichen politischen Wahrnehmung nicht direkt zusammen (vgl. Kap. 4.3.1).

[37] „Die politische Arbeit schafft letzten Endes Raum für Freie Software, der dann gefüllt wird durch diejenigen, die das tatsächlich umsetzen“ (Georg Greve im Interview am 25.6.2004 in Karlsruhe).

[38] http://www.slashdot.org

[39] http://www.newsforge.com

5.4 Zusammenfassung

Die Betrachtung der Strukturen des Feldes zeigt, dass sich innerhalb seiner gemeinsamen Grenzen ein hohes Maß an Heterogenität finden lässt. Dieses ergibt sich aus sehr unterschiedlichen Vorstellungen über Mitgliedschaft, die Größe der Projekte und ihren thematischen Fokus. Die Projekte bilden in sich relative geschlossene Orte mit jeweils eigenen Organisationsformen. Diese reichen von stark ausdifferenzierten und formalisierten Strukturen zu informelleren Formen, die eher auf Konvention, Aushandlung oder auch Zufall beruhen. Die Projekten weisen dabei unterschiedliche Entscheidungssysteme auf. Dabei ist zu beobachten, dass in allen Fällen die Gestaltung und Veränderung der Strukturen den Bedürfnissen der Produktion folgt. Neben den Projekten, die sich der Produktion widmen, existiert eine Reihe von Organisationen, die sich übergreifenden rechtlichen und politischen Fragen widmen. Zudem existieren Kommunikationsforen, die als Öffentlichkeit dem Austausch und der übergreifenden Diskussion dienen. Diese sind, wie auch die projektinternen technischen Kommunikationsmittel, von zentraler Bedeutung um die spezifischen Probleme des virtuellen Raums auszugleichen. Sie erleichtern auch soziale Prozesse in einer text-basierten Umgebung.

Kapitel 6

Prozesse der sozialen Kohäsion

Im folgenden Kapitel soll untersucht werden, wie die Projekte ihr Fortbestehen als soziale Zusammenhänge sichern. Dazu ist zunächst ein Blick auf die einzelnen Entwickler nötig. Die Betrachtung ihrer Lebenssituation, ihrer Ausbildung, ihres Berufes etc. soll einen Überblick über die Heterogenität bzw. Homogenität der Beteiligten geben. Da die Entwickler sich auf freiwilliger Basis beteiligen, sollen dann ihre Motive für ihre Mitarbeit aufgezeigt werden.

Im Folgenden soll untersucht werden, wie die Projekte ihre Strukturen mittelfristig stabil halten, um den Erfolg der Arbeit zu gewährleisten. Hierbei ist ein wichtiger Punkt der Umgang mit Konflikten. Darüber hinaus muss die Bindung der Einzelnen an die Projekte gesichert werden.

6.1 Persönliche Ebene der Entwickler

Die statistischen Daten, die in den folgenden Abschnitten betrachtet werden, stammen zum einen aus der FLOSS-Studie, einer der bekanntesten und umfangreichsten empirischen Studien im Feld (Ghosh et al. 2002) und aus meiner eigenen online-Umfrage.

6.1.1 Der private Hintergrund der Entwickler

An diese Stelle soll zunächst auf den persönlichen Hintergrund der Entwickler eingegangen werden. Als erstes springt ins Auge, dass der Anteil der Frauen nur 1 % bis 2 % beträgt, unabhängig vom geographischen Schwerpunkt der Studie (Ghosh et al. 2002: 8; Tab. 1[1]). Der Altersdurchschnitt der Entwickler beträgt knapp unter 30 Jahre, wobei er in Asien knapp über 30 beträgt (Shimizu et al. 2004: 4), in Europa

[1] Die Tabellennummern verweisen auf Anhang A

etwas darunter.[2] 75% bis 80 % sind in der Altersgruppe zwischen 20 und 30 Jahren. Es gibt einige wenige Entwickler, im Alter von 15 oder 16 Jahren und etwa 5 % sind über 40 Jahre alt (Ghosh et al. 2002: 9; Tab. 2). Gut die Hälfte der Entwickler hat eine feste Partnerschaft (Tab. 3). Unter diesen sind die Anteile der Verheirateten, der mit dem Partner zusammen Lebenden und der nicht mit dem Partner zusammen Lebenden etwa gleich groß (Ghosh et al. 2002: 11). Es sei darauf hingewiesen, dass ca. ein Viertel bei den Eltern leben. Ein knappes Sechstel der Entwickler hat mindestens ein Kind (Ghosh et al. 2002: 11; Tab. 4). Die Verteilung der Herkunft ist nur schwierig zu beurteilen, da die verschiedenen Studien jeweils (gezielt) andere regionale Schwerpunkte haben. Aus den allgemein angelegten Studien ergibt sich ein Übergewicht europäischer und nordamerikanischer Entwickler (Ghosh et al. 2002: 16f.; Tab. 5). Wenn man annimmt, dass dies nicht voll der Realität entspricht, sondern auf sprachliche Barrieren und unterschiedliche Erreichung der Zielgruppe zurückzuführen ist, so könnte man daraus zumindest auf ein in verschiedene regionale Bereiche geteiltes Feld schließen.

Der erreichte Ausbildungsstand der Entwickler ist hoch, etwa zwei Drittel verfügen über einen Universitätsabschluss, etwa die Hälfte dieser Gruppe sogar über einen Masters oder einen Doktortitel. Darüber hinaus nennen ein Viertel bis ein Drittel der Befragten einen für die Universität qualifizierenden Schulabschluss als höchste erreichte Ausbildungsstufe (Ghosh et al. 2002: 12; Tab. 6).[3] Rund ein Viertel der Entwickler studiert noch, über die Hälfte ist angestellt und gut 10 %sind selbstständig. Etwa zwei Drittel arbeiten im IT-Bereich, der weit überwiegende Teil davon als Programmierer (Ghosh et al. 2002: 13). Ein Drittel der Befragten wird in der einen oder anderen Form für die Entwicklung Freier Software bezahlt bzw. kann während der Arbeitszeit daran arbeiten (Ghosh et al. 2002: 65). Zu dem fällt auf, dass ein erheblicher Teil der Entwickler seinen Lebensunterhalt mit der Entwicklung proprietärer Software bestreitet (Ghosh et al. 2002: 26f.). Trotz der oben beschriebenen Wichtigkeit ideologischer Motive scheint es also kaum Berührungsängste zu geben (etwa 90 % geben dies auch auf direkte Nachfrage hin an) (Tab. 7).

Abgesehen von den noch im Studium Befindlichen und einigen wenige Ausnahmen verfügen die Entwickler über mittlere (22, 2 % mit US $ 1000 bis 2000) oder hohe (42 % mit über US $ 2000) Netto-Einkommen (Tab. 8). Um die internationale Vergleichbarkeit besser einschätzen zu können, wurde auch um eine subjektive Beurteilung der eigenen finanziellen Verhältnisse gebeten. Danach bewerten 80,5 % diese als gut oder sehr gut.

Knapp die Hälfte der Befragten gibt an, bis zu 5 Stunden pro Woche Freie Software zu entwickeln, allerdings wenden auch mehr als ein Viertel über 20 Stunden da-

[2]Die Fallzahlen der asiatischen Studie sind deutlich kleiner als die der anderen Regionen.

[3]Bei der hohen Zahl der unter 20-jährigen sollte man bedenken, dass diese ein Studium bzw. dessen Abschluss noch vor sich haben.

für auf. Dieser Personenkreis ist fast ausschließlich im IT-Bereich tätig (Ghosh et al. 2002: 22f.). Unter den Befragten überwiegen eindeutig Entwickler aus Projekten mit 10 und mehr Beteiligten (ca. zwei Drittel), nur etwa 15 % sind in Einpersonenprojekten tätig. Die Beteiligung an Großprojekten (30,9 %) und an Projekten mit um 10 Entwicklern (35,9 %) hält sich etwa die Waage (Tab. 10). Die Mehrzahl (62,4 %) der Entwickler ist gleichzeitig an mehreren Projekten beteiligt, zumeist an zwei bis fünf Projekten (57 %). Es überwiegen Entwickler, die an 5 oder weniger Projekten teilgenommen haben (71,9 %), nur ein verschwindender Anteil kann als sehr erfahren gelten (Ghosh et al. 2002: 31f.). Durchschnittlich beteiligen sich die Entwickler seit sechs Jahren an Freier Software, etwa die Hälfte ist schon länger dabei, unter 5 % sogar schon seit 15 oder mehr Jahren (Ghosh et al. 2002: 10). Etwa zwei Drittel der Entwickler haben in mindestens einem Projekt eine führende Funktion bekleidet.[4] Die Zahl derer, die in mehr als fünf Projekten eine solche Rolle innehatten liegt bei 7,2 %, etwa 0,5 % haben diese in elf oder mehr Projekten ausgeübt (Ghosh et al. 2002: 36). Die Altersverteilung in dieser Führungsgruppe variiert nur wenig gegenüber der Gesamtheit. Auch die Dominanz der im IT-Bereich beruflich Tätigen ist unter ihnen kaum ausgeprägter. Selbst die Länge des Zeitraums der Beteiligung an der Entwicklung Freier Software scheint keinen besonders großen Einfluss auf die Übernahme leitender Funktionen zu haben (Ghosh et al. 2002: 37ff.). Die Abweichung bei Personen, die erst relativ kurz dabei sind erklärt sich wohl damit, dass sie schlicht noch nicht genug Zeit hatten, um solche Positionen schon in mehreren Projekten innegehabt zu haben.

Das Feld weist einen hohen Grad an sozialer Homogenität auf. Die Entwickler sind also überdurchschnittlich gut ausgebildete junge Männer, die vor allem aus Europa und Nordamerika stammen. Freie Software ist für die meisten ein Hobby, für viele aber auch Teil ihrer beruflichen Arbeit. Man kann also einen hohen Grad an Professionalisierung im Feld feststellen. Für die Vergabe leitender Positionen in Projekten scheinen weder Alter noch Beruf, und auch die Erfahrung in der Entwicklung Freier Software eine hervorstechende Bedeutung zu haben. Diese Ergebnisse lassen sich dahingehend interpretieren, dass die Wissensschwelle für den Zugang zum Feld sehr hoch ist, aber innerhalb dieser Grenze formale Kriterien kaum noch eine Rolle spielen.

6.1.2 Motivation

Die Beteiligung an der Entwicklung Freier Software beruht auf Freiwilligkeit. Aus diesem Grunde kommt der Frage nach der persönlichen Motivation eine besondere Bedeutung zu, denn sie ist es, die das Phänomen Freie Software überhaupt erst

[4] Einpersonenprojekte sind hier allerdings nicht aufgeschlüsselt. Auch geht aus den Daten nicht hervor, ob Unterschiede zwischen großen und kleinen Projekten bestehen.

möglich macht. Die Motivation bezieht sich sowohl auf einzelne Projekte als auch auf die Idee der Freien Software insgesamt. Die Motivation ist der unterstützende Input, den die Projekte verarbeiten und aufrecht erhalten müssen, um ihr Fortbestehen zu sichern.

Mittlerweile gibt es zu diesem Thema einige wissenschaftliche Untersuchungen sowie Überlegungen von Entwicklern, die zu dem Ergebnis kommen, dass das Spektrum der Motivationen sehr vielfältig ist. Benno Luthiger identifiziert daraus sechs Kategorien (Luthiger 2004: 95ff.):

1. Gebrauch: Jemand entwickelt Software, die er selbst nutzen möchte.

2. Reputation und Signalwirkung: Die These der Reputation geht auf Eric Raymond zurück, der argumentiert, dass Entwicklern, die schon alle materiellen Dinge besitzen,einen höheren Status nur noch durch Geben erreichen können (Raymond 2000b). Die Gaben der Entwickler Freier Software beruhen auf *Freiwilligkeit.* Einzelne Entwickler mögen sich zwar verpflichtet fühlen, etwas für die ganze Software, die sie nutzen zurückzugeben, aber es gibt keine Kontrollinstanz (z. B. eine Öffentlichkeit), die dieses überprüfen könnte. Wenn man nichts oder nur wenig an die Allgemeinheit gibt, bleibt man einfach nur unbemerkt. Wenn man dagegen viel (quantitativ und qualitativ) gibt, kann man hohes Ansehen erlangen. Es ist bei den Entwicklern Freier Software also so, dass durch das Geben zwar eine positive Reputation, durch Nicht-Geben aber keine negative erlangen werden kann.[5]
   Es besteht die Möglichkeit für die Entwickler ihre Reputation zu monetarisieren, indem potentielle Arbeitgeber durch das Wissen über die Tätigkeit eines bestimmten Entwicklers Rückschlüsse auf sein Talent ziehen können. So kann die interne Reputation eine Signalwirkung außerhalb des Felds Freier Software entfalten (Lerner/Tirole 2000: 14ff.).

3. Identifikation mit der Gruppe: Karim Lakhani und Robert Wolf haben in ihrer Untersuchung festgestellt, dass ein Teil der Motivation durch die Identifikation der Entwickler mit ihrem Projekt entsteht. Zu dieser Aussage kamen sie durch die höhere Projektbindung von freiwilligen im Gegensatz zu bezahlten Entwicklern (Lakhani/Wolf 2003: 23).

4. Lernen: Freie Software bietet die Möglichkeit die eigenen Programmierfähigkeiten zu verbessern (Gosh et al. 2002: 45). Dabei können die Projekte die Gestalt

[5] Raymond vergleicht die Entwickler Freier Software mit dem Kwakiutl und ihrem Potlach, einem Zeremonialtausch. Es ist fraglich, ob Raymond bei der Wahl seines „anthropologischen" Beispiels ein gutes Händchen hatte. Denn der Potlach hatte einen Zwangscharakter. Den Ausführungen von Marcel Mauss zum Potlach folgend stellt man fest, dass derjenige, der den Potlach nicht erwidern konnte, sein Gesicht verlor und mit Schuldknechtschaft sanktioniert wurde (Mauss 1990: 101). Solche Folgen haben die Gaben bzw. Nicht-Gaben der Entwickler nicht.

einer Spielwiese haben – man probiert Verschiedenes aus, macht Fehler, lernt aus ihnen und hat irgendwann Erfolgserlebnisse.

5. Altruismus: Hierunter fasst Luthiger ideologische Beweggründe, also den Wunsch für die Freiheit der Information und der Software zu kämpfen, weil man diese für ein hohes Gut hält. Daneben identifiziert Luthiger eine zweiten Form von Altruismus – eine als gerecht empfundene Reziprozität (den Wunsch für die selbst genutzte Software etwas zurückzugeben).[6]

6. Spaß: Die Freude an der Tätigkeit des Programmieren ist ein nicht zu unterschätzender Beteiligungsgrund. Dies hebt auch Linus Torvalds für seine eigene Person hervor und betitelte seine Autobiographie dementsprechend mit „Just for Fun“ (Torvalds 2001a; 2001b: 15).

Luthiger weist darauf hin, dass die verschiedenen Motivationsarten sich ergänzen, dass aber bisher eine Abschätzung der Relevanz der verschiedenen Typen fehlt (Luthiger 2004: 95; 98). Dieses soll im folgenden in Ansätzen versucht werden. Dazu soll auf die Ergebnisse meiner eigenen Umfrage sowie die der FLOSS Studie zurückgegriffen werden (Ghosh et al. 2002).

Es ist zu bedenken, dass einige Motive sich erst im Laufe der Teilnahme entwickeln können. Dazu gehören beispielsweise die Erlangung von Reputation, die persönliche Bindung an das von einem selbst geschaffene Produkt („das ist mein Baby“) und an das Projekt und auch arbeitsethische Vorstellungen („ich muss das Programm fertig schreiben“). Diese Art der Motivation, die sich erst in der Wechselwirkung mit der Beteiligung herausbildet, möchte systemische Motivation und von der Einstiegsmotivation unterscheiden.

Zunächst ist zu diesem Themenkomplex zu prüfen, welche Motivationen für die Entwickler zum Einstieg eine Rolle spielen. Die Ergebnisse der FLOSS Studie zeigen, dass „learn and develop new skills“ (78,9 %) und “share knowledge and skills“ (49,8 %) die wichtigsten Gründe zum Einstieg waren. Danach folgen „participate in a new form of cooperation“ (34,5 %), „improve OS/FS products of other developers“ (33,7), „participate in the OS/FS scene“ (30,6 %), „think that software should not be a proprietary good“ (30,1 %) und „solve a problem that could not be solved by proprietary software“ (29,7 %) (Ghosh et al. 2002: 45). Man kann diese Motive grob in vier Kategorien unterteilen: Eigeninteresse (Verbesserung und Ausprobieren der eigenen Fähigkeiten), ideologische Motive (bezügl. der Freiheit der Software), technisch-praktische Motive (Wunsch nach neuer bzw. besserer Software) und schließlich soziale Motive (Wunsch, Teil einer kooperierenden Gruppe zu sein).

[6]Während der altruistische Charakter der ideologischen Beweggründe im Sinne einer „Weltverbesserung“ nachvollziehbar ist, ist er beim Begriff der Reziprozität zu bezweifeln. Diese stellt meist ein klar definiertes kalkulierbares und mit Sanktionen verbundenes (unbewusstes) Zwangsverhältnis dar Spittler 1980: 18).

Andere Beweggründe spielen kaum eine Rolle. Aus den zitierten Daten ergibt sich, dass Eigeninteresse und soziale Motive im Vordergrund stehen. Nach meiner eigenen Untersuchung werden technisch-praktische Motive als besonders wichtig angesehen. Die Ziele, ein Programm zu verbessern oder ein Problem zu lösen, werden mit durchschnittlich 1, 94 bzw. 1,92 (Skala von 1 bis 4) bewertet. Der Beweggrund mit dem besten Wert ist allerdings „programming seemed to be fun“ (1,74) (Tab. 11). Vergleicht man die Ergebnisse, zeigt sich, dass Eigeninteresse und technisch-praktische Motive überwiegen, soziale und ideologische Motive folgen mit einigem Abstand. Die bewusste Suche nach Reputation ist nur von untergeordneter Bedeutung.[7]

Bei der Betrachtung der systemischen Motivation zeigt sich, dass die Ordnung der einzelnen Kategorien im wesentlichen die gleiche bleibt. Es lässt sich aber eine deutlich wachsende Bedeutung der ideologischen Motive feststellen. So erhöhen sich die entsprechenden Werte in der FLOSS Studie um ein Drittel bis die Hälfte, in meiner Untersuchung bestätigt sich die Entwicklung (Ghosh et al. 2002: 45; Tab. 12). Dort zeigt sich auch eine steigende Relevanz der sozialen Motive. Dies deutet daraufhin, dass im Laufe der Mitarbeit soziale Prozesse in den Projekten ein Eigengewicht gewinnen. Die Stärkung der ideologischen Beweggründe weist auf eine größere Identifikation mit Freier Software hin.

6.2 Zusammenhalt auf Projektebene

Die Projekte müssen als Zusammenschlüsse von Freiwilligen Mechanismen entwickeln, um ihre Stabilität zu sichern. Diese sind einerseits problemlösend, wenn es um die Bearbeitung konkreter Konflikte geht. Andererseits sind sie eher präventiver Natur, indem sie die Mitglieder an das Projekt binden und ihnen seine Kultur vermitteln.

6.2.1 Konflikte

Zunächst werden die verschiedenen Konfliktfelder, die im Bereich Freier Software existieren vorgestellt. Im Anschluss werden die Möglichkeiten der Konfliktbearbeitung besprochen werden. Schließlich sollten möglichen Konfliktfolgen untersucht werden.

6.2.1.1 Konflikttypen

Im Bereich Freier Software sind zwei Kategorien von Konflikten zu beobachten: interne und grenzüberschreitende. Die internen Konflikte bestehen zwischen den Mitgliedern des Feldes und teilen sich in drei Typen ein: ideologische, technische und

[7] Unterschiedliche Fragetypen (Auswahl der vier wichtigsten Motive aus vierzehn Möglichkeiten bei der FLOSS Studie, Bewertung der Wichtigkeit von zwölf Motiven auf einer Skala von eins bis vier) und Unterschiede bei den angebotenen Alternativen müssen beim Vergleich der Daten berücksichtigt werden.

personelle. Die grenzüberschreitenden Konflikte finden zwischen Mitgliedern des Feldes und Außenstehenden, bzw. Mitgliedern, die durch ihre Handlungen sich zu Außenstehenden machen, statt. Zu dieser Kategorie gehören die zwei Typen kultureller und rechtlicher Konflikt. Natürlich sind die einzelnen Fälle in der Realität meist eine Mischung aus den verschiedenen Typen.

Ideologische Konflikte beziehen sich meist auf unterschiedliche Vorstellungen über und Herangehensweisen an die Freiheit der Software. So ist Gnome beispielsweise durch die Auseinandersetzung um KDE und die dafür verwendete, damals unfreie Bibliothek Qt von Trolltech entstanden. Ebenso ist die Entstehung von Open Source hier zu verorten ist. Der Begriff wurde eingeführt um sich von der FSF zu distanzieren (vgl. Kap. 4.3.1).

Technische Konflikte betreffen unterschiedliche Auffassungen, wie ein technisches Problem am besten gelöst wird. Sie haben ein großes Konfliktpotential – es sein an das ausgeprägte Streben nach technischer Perfektion erinnert. Ein bekanntes Beispiel ist die Auseinandersetzung über die Vor- und Nachteile von Mikro- oder Makrokernels. Dieser Streitpunkt war ein Aspekt des Disputs zwischen Andrew Tanenbaum und Linus Torvalds zu Beginn von Linux (o.V. 1999). Die Frage, die sich bei diesen Konflikten stellt, ist, was der beste Weg ist, um etwas bestimmtes zu erreichen.

Persönliche Konflikte basieren auf dem Verhalten Einzelner, die einen Normbruch begehen. Ein Beispiel ist Theo de Raadt, der aus NetBSD ausgeschlossen wurde, da ihn die anderen offenbar für sozial nicht mehr tragbar hielten, auch wenn sie seine Programmierleistungen sehr schätzten (Glass 1994). Es fragt sich, ob die Gründung von Open Source nicht zum Teil auch auf einem persönlichen Konflikt zwischen Eric Raymond (und vielleicht auch anderen Personen) und Richard Stallman basiert. So lässt sich in der Stallman-Biographie folgendes dazu finden: „Active in the GNU Project during the 1980s, Raymond had distanced himself from the project in 1992, citing, like many others before him, Stallman's ‚micro-management' style. ‚Richard kicked up a fuss about my making unauthorized modifications when I was cleaning up the Emacs LISP libraries,' Raymond recalls. ‚It frustrated me so much that I decided I didn't want to work with him anymore.' (Williams 2002: 156).[8]

Kulturelle Konflikte betreffen weniger eine Art von Normverstöße als das Nicht-Teilen der Normen. Einer dieser kulturellen Konflikte entstand in den letzten Jahren durch die starke Verbreitung Freier Software. Mittlerweile gibt es viele User, die nur über mäßige Computerkenntnisse verfügen. Aber nicht nur ihr Wissenstand ist geringer – den sie ja aufholen könnten – auch ihre Einstellung ist eine andere. Die Erwartungshaltung gegenüber den Entwicklern hat viel konsumhaftere Züge angenommen. Man wartet auf ein bestimmtes Feature, anstatt sich selbst zu beteiligen,

[8]Sam Williams gibt leider keine Quellenangabe zu dem Zitat von Eric Raymond an. Glyn Moody beschreibt aber ähnliche Konflikte zwischen Stallman und dem Debian Projekt, das für kurze Zeit von der FSF finanziell unterstützt wurde (Moody 2001: 127ff.).

oder es werden den Entwicklern Fragen gestellt, die eindeutig in der Dokumentation beantwortet werden. Diese Haltung ist vergleichbar mit der eines Konsumenten eines kommerziellen Produktes. So kommt es auch immer wieder zu Auseinandersetzungen zwischen Entwicklern und Usern, die nicht mehr der (Sub-)Kultur der Freien Software angehören.

Den letzten Konflikttyp möchte ich als rechtlichen Konflikt bezeichnen. Sie beziehen sich auf Verstöße gegen das Urheber- und das Vertragsrecht. So kopierte ein Mitarbeiter von WorldForge unrechtmäßig ein Musikstück und gab es als sein eigenes aus. Ein weiterer Fall ließ sich bei Linux finden, dort beanspruchte zwar niemand die Urheberschaft unberechtigterweise, aber missinterpretierte die GLP auf eine kritische Weise.[9] Der berühmteste Fall dürfte allerdings Caldera/SCO-Group sein (vgl. Kap. 4.2.2). Es gibt aber auch Lizenzverstöße durch Akteure, die sich nie im Bereich der Freien Software befunden haben, insbesondere Unternehmen. Diese Verstöße beziehen sich insbesondere auf Lizenzen mit einem Copyleft, bei denen die Unternehmen den Quellcode ihrer auf GPL-Software basierenden Programme nicht veröffentlichen.[10]

6.2.1.2 Konfliktbearbeitung

Die Möglichkeiten der Konfliktbearbeitung sind hauptsächlich durch vier Faktoren bestimmt: die räumlichen Gegebenheiten der sozialen Zusammenschlüsse, die Freiwilligkeit der Beteiligung, die jeweiligen politischen Systeme der Projekte und den Konflikttyp. Aufgrund der physischen Distanz und der Beschränktheit auf Text, stehen körperliche Lösungen nicht zur Verfügung. Über Attacken auf Rechner und Daten von anderen habe ich nie etwas gehört – auch wenn die meisten wohl die technischen Fähigkeiten dazu hätten. Auch wenn jedes Jahr auf dem Linuxtag in Karlsruhe ein Hacking Contest (im Sinne der Cracker) veranstaltet wird, dienen solche Veranstaltungen dem Demonstrieren des technischen Wissens. Es ist davon auszugehen, dass wenn es öffentlich bekannt würde, dass ein Entwickler in den Rechner eines anderen einbräche, er jedes Ansehen seiner Peers verlieren würde. Das Gros der Entwickler beteiligt sich freiwillig und ist durch keinerlei Verträge etc. an seine Tätigkeit gebunden. Sie können daher jederzeit ihre Arbeit einstellen – ohne das dies weitere Konsequenzen hat, als dass sie womöglich die soziale Beziehungen zu ihren Mitentwicklern aufgeben müssen.

Den Zusammenhang von politischen Systemen und Streitregelungsmöglichkeiten haben schon Gerd Spittler (1980) und Trutz von Trotha (2000) aufgezeigt. Spittler

[9] Vgl. http://marc.theaimsgroup.com/?l=linux-kernel&m=109171462620815&w=2; http://marc.theaimsgroup.com/?l=linux-kernel&m=109187409710377&w=2.

[10] Es kommt zwar auch zu Verstößen gegen die BSL-Lizenzen durch die Nicht-Nennung der ursprünglichen Autoren, die wird allerdings nicht so ernst genommen. Interview mit einem NetBSD Mitarbeiter am 5.4.2003.

unterscheidet zunächst zwischen einfacher Streitregelung und solcher durch organisierte Rechtsinstanz. Die einfache Streitregelung teilt sich in Verhandlungen und einfache Rechtsinstanz auf. Die Verhandlungen können direkt zwischen den Kontrahenten stattfinden oder unter Hinzuziehung von Vermittlern. Einfache Rechtsinstanzen zeichnen sich durch die Existenz von Richtern aus. Organisierte Rechtsinstanzen lokalisieren sich in Gerichten, zu denen eine entwickeltere Rollendifferenzierung, zwischen Richter, Anwälten, Staatsanwälten und Apparate für Voruntersuchungen und für Urteilsvollstreckungen gehören (Spittler 1980: 7). Verhandlungen sind nicht nur durch die Normen beeinflusst, sondern auch durch das Machtverhältnis der Konfliktparteien. Die Lösung des Konflikts hat eher die Gestalt einer Übereinkunft, als die einer Entscheidung aufgrund der Anwendung abstrakter Regeln. So ist das primäre Ziel die Versöhnung der Parteien und weniger, eine abstrahierte Form der Gerechtigkeit walten zu lassen. Bei Verhandlungen können Vermittler hinzugezogen werden, die im Gegensatz zu Richtern nicht über die Kompetenz verfügen, Entscheidung zu fällen (ebd.: 8), d .h. bei Vermittlern verbleibt die Verfügungsmacht bei den Streitparteien /Trotha 2000: 334). Auch bei einfachen Rechtsinstanzen steht die Versöhnung, hier durch Schlichtung, im Vordergrund. Die Richterrolle ist nicht mit der in einem staatlichen Gefüge vergleichbar, meist ist sie nur eine Unterrolle eines Häuptlings oder „Bürgermeisters“. Häufig haben die Richter auch keine spezifische Ausbildung für ihr Amt (Spittler 1980: 11). Sie sind aber Teil der selben rechtlichen Subkultur wie die Streitparteien und die Bevölkerung, d. h. sie haben die gleichen Rechtsvorstellungen (ebd.: 13). Trutz von Trotha zeigt die Möglichkeit der Rechtsinstanz Druck auf die Kontrahenten ausüben zu können,so dass sie vor ihr erscheinen müssen. Dieser Druck muss keinen Zwangscharakter haben, sondern er kann in der Reputation der Richter begründet sein, d.h. dass man sich der Rechtssprechung nicht verweigern kann, wenn man sich nicht im sozialen Umfeld höchst unbeliebt machen will (Trotha 2000: 333). Gerd Spittler weist daraufhin, dass in Gesellschaften ohne Herrscherinstanz Selbsthilfe bzw. Selbstjustiz das grundlegende Mittel zur Rechtsdurchsetzung ist, ein Zustand der trotzdem die Möglichkeit von Verhandlungen lässt. Allerdings werden die Verhandlungen in so einer Konstellation immer durch das Wissen über die existierende Möglichkeit der Selbsthilfe stark beeinflusst (Spittler 1980: 21f.). Ebenso wirkt sich die Existenz von staatlicher Gesetzgebung und Gerichtssystemen auf die Streitregelung der lokalen, akephalen Gesellschaft aus (vgl. ebd.: 23ff.).

Im Feld Freier Software lassen sich verschiedene Streitregelungstypen finden. Im folgenden sollen sie und die in ihnen konkret auftretenden Konfliktbearbeitungen beschrieben werden.

Zum einen gibt es Projekte mit formalen Prozeduren. Hierfür braucht es ein kodifiziertes System in dem die Verfahren festgelegt sind. Formale Prozeduren sind mir bislang nur von Debian bekannt. Dort haben die Entwickler die Möglichkeit, wenn

über wichtige Fragen anders kein Konsens erreicht wird, eine Wahl durchzuführen.

In Projekten mit einer anerkannten Autorität (durch Wahl oder Tradition, z.B. der Projektgründer) kann es bei Konflikten eine Art Richter geben. Dies kann der Projektleiter, aber auch ein Maintainer sein. Diese Person kann entweder ein Machtwort sprechen oder einfach eine Partei ignorieren. Es ist auch möglich, dass sie eine allgemeine Abstimmung ausruft. Der Erfolg dieses Mechanismus hängt stark von der jeweiligen Projektstruktur ab, außerdem muss die Machtperson bei ihrem Verhalten in Konfliktsituationen bedenken, auf welche Weise die eigene Macht legitimiert ist. Wenn es sich um eine rein charismatische Herrschaft handelt, kann sie in solchen Situation nicht sehr weit gehen, ohne dass sie ihr eigene Position gefährdet.

In Projekten ohne zentrale Autorität müssen Konflikte durch Verhandlungen bearbeitet werden. Die Formen sind hierbei vielfältig, sie können von einer rationalen, sachlich begründeten Argumentation bis zu persönlichen Beleidigungen (Flames) reichen. Als Konfliktmittel kann hier auch die Zuweisung von Schande (negative Reputation) fungieren (Elwert 2001: 17). In Fällen der Aushandlung kann man sich friedlich einigen (einen Konsens finden) oder eine der beiden Parteien gibt irgendwann auf.

In den Projekten werden aber nur interne Konflikte wie eben beschrieben bearbeitet. Die Behandlung von grenzüberschreitenden Konflikten orientiert sich an dem Konflikttyp und ist vom politischen System der Projekte unabhängig. Bei kulturellen Verstößen werden Außenstehende zunächst einmal daraufhin gewiesen und erhalten eine Belehrung über die richtige Art des Umgangs. Bei wiederholten Verstößen der gleichen Person, wird diese durch technische Filter von den Kommunikationsmedien vom Projekt fern gehalten. Bei gleichartigen Verstößen durch verschiedenen Personen, findet eine Anpassung der Kommunikationsmedien der Projekte statt. Es wird entweder die Mailingliste gewechselt oder auch ein IRC-Channel für Nutzerfragen geschlossen.

Bei rechtlichen Konflikten werden die Täter zunächst ebenfalls auf die Falschheit ihrer Handlungen hingewiesen. Zeigen sich die Personen (oder Unternehmen) uneinsichtig, werden sie im Falle von Noch-Dazugehörigen des Projekts und bei hohem Bekanntheits- und Schweregrad des Verstoßes des Feldes Freier Software verwiesen. Bei Lizenzverstößen werden zusätzlich weitere Maßnahmen getroffen. Der Copyrightinhaber oder sein rechtlicher Vertreter, dies kann die FSF, die Projektstiftung oder jeder Rechtsanwalt sein, weisen die Person oder das Unternehmen noch einmal auf ihr Fehlverhalten hin. Wenn letztere immer noch nicht einlenken, kommt es zur Klage. Diese Möglichkeit ist erstmals in diesem Jahr angewendet worden (vgl. Kreutzer 2004). Der Gang zu staatlichen Gerichten bedeutet aber ein Verlassen des Feldes Freier Software. Die organisierte Streitregelung spielt also auch eine Rolle, allerdings findet sie nicht mehr innerhalb des Feldes statt. Allenfalls die erste rechtliche Prüfung kann als Teil der formalen Voruntersuchung angesehen werden.

6.2.1.3 Konfliktfolgen

Im vorigen Abschnitt wurden die verschiedenen Möglichkeiten der Konfliktbearbeitung im Feld Freier Software vorgestellt. Nun soll auf die Folgen der Konflikte eingegangen werden. Auch wenn Rechtsinstanzen in einigen Projekten vorhanden sind, ist die mögliche Sanktionierung nur sehr begrenzt. Die härteste Sanktion, die eigentlich nur bei persönlichen und kulturellen Konflikten Anwendung findet, ist der Ausschluss einer Person. Er geschieht dadurch, dass jemand sein SCM-Schreibrecht verliert und von den Mailinglisten u.ä. technisch boykottiert wird (dies ist bei NetBSD mit Theo de Raadt und mit einer Person bei WorldForge geschehen). Diese Möglichkeit wird nur bei gravierenden Verstößen angewendet, und ist sehr selten. Eine abgeschwächte Form der Sanktionierung ist das Ignorieren einer Person. Außerdem gibt es immer die Möglichkeit der Zuweisung von Schande. Das kann zu einer negativen Reputation der betreffenden Person führen – hierbei ist zu bedenken, dass Reputation ein wichtiges strukturierende Element innerhalb der Projekte ist.

Es ist also so, dass man eher jemanden hindern kann, eine Sache weiter zu tun, als dass man jemand zwingen kann, etwas zu tun, da es hierfür kaum angemessene Mittel gibt.

Neben einer Sanktionierung gegen einzelne können Konflikte auch Folgen für das ganze Projekt haben. Einzelne Mitarbeiter können ihre Tätigkeit einstellen, da in ihrer persönlichen Kosten-Nutzen-Rechnung die Kosten nicht mehr tragbar sind. Außerdem kann es zu einer Spaltung des Projekts kommen. Bei einem Fork (Gabelung) spaltet sich ein Projekt in parallele Stränge auf, die nur noch bedingt kompatibel sind. Forks sind aufgrund der Lizenzen immer möglich und diese Möglichkeit wird auch explizit verteidigt.[11] Es ist möglich, dass Forks nur temporär sind und, dass nachdem sich eine Variante in ihrer weiteren Entwicklung als deutlich erfolgreicher dargestellt hat, die andere wieder mit der erfolgreicheren Lösung fusioniert. Temporäre Forks stellen aber nur bei technischen Konflikten einen Lösungsweg dar. Im Falle eines Forks haben Richter keinen Einfluss, Vermittler möglicherweise schon. Ihr Erfolg hängt vom Willen der Konfliktparteien zu Kooperation und Versöhnung und von der Fähigkeit und dem Ansehen des Vermittlers ab.

Forks und der Austritt einzelner Personen sind Exitoptionen, wenn der Widerspruch – also die Konfliktregelung in der beschriebenen Form – keinen Erfolg hat (Hirschman 1974).

[11] Vgl. Beiträge der Linux-Kernel Mailingliste: http://marc.theaimsgroup.com/?l=linux-kernel&m=108757316101769&w=2, http://marc.theaimsgroup.com/?l=linux-kernel&m=108757316229168&w=2, etc.

6.2.2 Stabilisierung

Über die Lösung konkreter Konflikte hinaus sind stabilisierende Maßnahmen für den Erfolg von Projekten entscheidend – für ihre Existenz als soziale Zusammenschlüsse und so auch für die eigentliche Entwicklungsarbeit.

Ein zentrales Mittel ist hierzu die Enkulturation, durch die vorhandenen und vor allem neuen Mitgliedern sowohl die Werte und Normen, als auch die Arbeitsweise und Projektprozesse vermittelt wird. Viele Projekte haben einführende Texte auf ihrer Webseite, wie das Projekt technisch und kulturell funktioniert. Darüber hinaus gibt es auch, vom Feld der Freien Software unabhängig, viele Guidelines, wie man sich beispielsweise auf Mailinglisten oder auch in IRC Channels verhalten soll. Ein wichtiger Anteil hat hier die Nachwuchsförderung. Viele Neue geben als „Newbies" zu erkennen und erhalten daraufhin von erfahreneren Entwicklern unterstützende Hinweise zu ihrem Code und sie werden auf Regelverstöße hingewiesen. Dies passiert entweder durch persönliche Anleitung oder sie werden auf Quellen, in denen die betreffenden Regelungen aufgeführt sind, hingewiesen. Diese Tutorfunktion wird auch von Entwicklern übernommen, die selbst erst relativ kurz im Projekt sind – was wiederum zu deren eigener Bindung an das Projekt beiträgt. Außerdem gibt es Einsteigertätigkeiten, die zwar nicht nur von Neuen erledigt werden, also Aufgaben, von denen aus man gut sich weiter orientieren kann. Solche Tätigkeiten sind beispielsweise Betatesten (man untersucht eine neue Version gezielt nach Fehlern) oder auch Übersetzen oder Dokumentation. Die Enkulturation ist meist nicht formalisiert.

Die Enkulturation setzt sich über die Phase des Projektbeitritts hinaus fort. Zudem erfolgt ein gradueller Aufstieg in der Hierarchie des Projektes. Wichtige Stufen hierbei sind etwa der Erhalt einer Projekt-Emailadresse und ein eigener Zugang zum SCM. Diese Schritte haben eine hohen symbolischen Gehalt, weshalb man sie als Teil eines integrierenden Rituals deuten kann.

Ein besonders ausgeprägtes Beispiel ist der New Maintainer Process von Debian. Eine Person, die offizieller Entwickler (Debian Developer) werden will muss ihn durchlaufen. Diese Position markiert bei Debian so etwas wie die Vollmitgliedschaft – so dürfen beispielsweise nur die Debian Developer an den Wahlen im Projekt teilnehmen. Der Prozess hat fünf Stufen und wird durch einen Application Manager begleitet. Als erstes stellt der Aspirant einen Antrag. Als zweiten Schritt musss er seine Identität beweisen – dies wird durch das Erstellen eines GPG-Schlüssels[12] und das anschließende Signieren durch einen anderen Debian Developer erledigt. Im dritten

[12]GPG ist ein Programm zur Verschlüsselung von Kommunikation im asymmetrischen publickey Verfahren. Dabei hat jeder Benutzer zwei Schlüssel, also längere Zahlencodes. Den einen gibt er bekannt, mit ihm werden Botschaften an ihn verschlüsselt. Diese sind jedoch mit dem public key nicht wieder entschlüsselbar, sondern nur mit dem private key, den der Benutzer geheim hält. Als Signieren von public keys wird die Bestätigung der Identität des Ausgebers eines public key bezeichnet, die natürlich immer physische Präsenz erfordert.

Schritt geht es um die Philosophie und Prozeduren von Debian. Der Bewerber muss in einem Diskussionsprozess mit dem Application Manager unter Beweis stellen, dass er den Debian Social Contract kennt und dessen Prinzipien teilt. Die Dokumentation dieses Gesprächs wird vom New Maintainer Committee bewertet. Fällt dessen Urteil positiv aus, muss der Neuling noch im vierten Schritt seine technischen Fähigkeiten nachweisen, indem er bestimmte Programmieraufgaben löst oder zur Dokumentation oder zur Infrastruktur beiträgt. Der Application Manager legt die Aufgaben fest und diese müssen vom New Maintainer Committee abgesegnet werden. Als fünften und letzten Schritt schickt der Application Manager einen Bericht an das Komitee, in dem er auch eine begründete Empfehlung ausspricht. Nach einer letzten Prüfung des New Maintainer Committees und eventuell weiteren Fragen an den Kandidaten wird dann von den Developer Accounts Managers ein Zugang zum Debian SCM für den Neuling eingerichtet. Schließlich wird die Neuaufnahme über die Mailinglisten bekannt gegeben.

Der New Maintainer Process stellt also ein richtiges Initiationsritual dar. Alle drei von van Gennep (1999: 21) identifizierten Phasen eines Übergangsritus – Trennungsriten, Umwandlungsriten und Angliederungsriten – lassen sich hier finden. Die Trennung besteht darin, dass die Bestätigung der Identität den Kandidaten aus der Masse der anonymen Projektmitglieder (die nicht den Status eines Debian Developer haben) herauslöst. Die Umwandlung geschieht durch die Einführung in die Philosophie und die Kontrolle der Fähigkeiten. Die Angliederung ist dann die Eröffnung eines Zugangs zum SCM. Die Bekanntgabe der Aufnahme ist die öffentliche Reintegration in das Projekt.[13] In anderen Projekten ist der Erwerb eines SMC-Zugangs etc. zwar auch an Regeln gebunden und auch hier könnte man die erwähnten drei Phasen auffinden. Aber das Verfahren ist bei weitem nicht so stark differenziert, so dass der symbolische Gehalt weniger sichtbar ist.

Projekte bilden auch eine eigene Identität heraus. Sie wird häufig auch mit Hilfe von Symbolen wie Maskottchen verstärkt. Bei realweltlichen Treffen finden sie sich auf T-Shirts, Tassen und in Stofftierform, um die Bindung an das Projekt nach innen und die Abgrenzung außen zu signalisieren. KDE organisiert als Integrationsmaßnahme jedes Jahr beim Karlsruher Linuxtag eine Party, die vor allem dem sozialen Austausch dient (und zu dem 2004 auch Mitglieder von Gnome und Debian eingeladen wurden). Einige Projekte führen manchmal Arbeitstreffen durch, bei denen aber auch soziale Prozesse wegen der physischen Präsenz möglich sind, die online nicht vorkommen können. Aber auch allgemeine Messen etc. tragen sehr zum Teamgefühl der einzelnen Projekte bei.

Die freundschaftliche Konkurrenz untereinander, gut zu beobachten zwischen KDE

[13]Im Unterschied zu anderen Initiationsriten bleiben die Kandidaten während der Umwandlungsphase Teil des Projekts, sie können weiter ihre vorherige Rolle spielen und unterliegen keinen besonderen Beschränkungen ihres Verhaltens (vgl. van Gennep 1999: 70ff.).

und Gnome Entwicklern, verstärkt auch das Wir-Gefühl in den einzelnen Projekten.

Eine besondere Form der Stabilisierung stellt die Tabuisierung bestimmter negativer Vorgänge dar, die das Projekt gefährden würden. Hierunter fallen besonders Forks. Sie würden nur vorkommen, wenn es gar nicht mehr anders geht und würden immer mit öffentlichen Selbstvorwürfen der Spalter einhergehen (Raymond 2000b). Hintergrund ist die weit verbreitete Auffassung, dass Forks sich letztlich negativ für die weitere Entwicklung auswirkt, da sich die personellen Ressourcen aufspalten. Dies scheint ein Hinweis auf eine Mangel-Ökonomie (an Zeit) zu sein. Dieser Mangelzustand wird durch Kooperation ausgeglichen (vgl. Elwert 1980).

Schließlich gibt es noch die öffentliche Sphäre, die es möglich macht, dass nicht nur Projektmitglieder, sondern auch andere Entwickler verfolgen (können), was in einem bestimmten Projekt passiert. Es ist anzunehmen, dass so ein sozialer Druck und damit eine soziale Kontrolle von außen entstehen. Der Tratsch auf Treffen wie dem Linuxtag bestätigt diesen Eindruck. Dort herrschen immer bestimmte Themen vor, die die meisten Personen beschäftigen, beispielsweise 2002 die recht neu gegründete Free Software Foundation Europe oder die Änderung des Debian Social Contract im Jahre 2004. Diese wird durch interne Magazine (Linux-Magazin), und spezielle Webseiten (z.B. Slashdot, Newsforge) gestärkt.

6.3 Zusammenfassung

Die nähere Betrachtung der Entwickler im Feld Freie Software zeigt, dass es unter ihnen einen hohen Grad an Homogenität gibt. Die soziale und berufliche Situation der einzelnen weisen ebenso wie ihr Geschlecht und ihr Alter wenig Varianz auf. Es lässt sich ein hohes Maß an Professionalität finden, viele Entwickler betreiben dieses Hobby sogar als Teil ihres Berufes. Innerhalb dieser homogenen Gruppe spielen dann formale Merkmale anscheinend nur noch eine geringe Rolle. Die Entwickler begründen ihr Engagement vor allem mit technischen und ideologischen Motiven, besonders der Wunsch, die eigenen Fähigkeiten anzuwenden und zu verbessern hat eine wichtige Bedeutung, aber auch einfach die Freude am Programmieren. Trotz der herausragenden Bedeutung einzelner Motive sind die Gründe für die Mitarbeit sehr vielfältig. Diese Motive sind relativ stabil, trotzdem lässt sich aber ein gewisser Enkulturationsprozess daran ablesen, dass die ideologischen Motive noch gestärkt werden und auch soziale Aspekte, etwa sich gerne als Teil eines Projektes zu sehen, wichtiger werden. Es scheint den Projekten also zu gelingen, die Freiwilligen so anzusprechen, dass ihre Motivation eher noch zunimmt.

Um dies zu erreichen müssen die Projekte Methoden entwickeln, mit denen sie einem Auseinanderbrechen bzw. einem Rückzug einzelner vorbeugen und Konfliktfälle lösen. Konflikte innerhalb der Projekte können persönlicher, ideologischer und

technischer Natur sein. Dazu kommen Konflikte, die Determinanten des Feldes insgesamt betreffen, aber im Projekt ausgetragen werden. Dazu gehören kulturelle und rechtliche Auseinandersetzungen (wobei letztere teilweise außerhalb des Feldes bearbeitet werden). Wie im Falle der Entscheidungsstrukturen lassen sich auch zur Konfliktlösung eine Vielzahl formeller und informeller Formen finden. Bei rechtlichen Konflikten kann bzw. muss die Lösung an (staatliche) Institutionen außerhalb des Feldes delegiert werden. Schlichtung und Integration stehen im Vordergrund, da eine konfrontative Auseinandersetzung die Stabilität und die weitere Arbeit des Projektes in Frage stellen könnte. Der Rahmen möglicher Sanktionen ist durch die räumlichen Umstände und die Freiwilligkeit sehr eng. Er umfasst im wesentlichen defensive Mittel, Möglichkeiten, jemanden zu einem bestimmten positiven Verhalten zu zwingen existieren nicht. Sanktionen müssen auch immer gegen die negativen Folgen für das Projekt – bis hin zur Teilung durch einen Fork – abgewogen werden. Um Konflikte von vornherein zu vermeiden, bemühen sich die Projekte ihren Mitgliedern die Normen und Werte der Freien Software nahe zu bringen. Neben dieser Übereinstimmung produzierenden Enkulturation spielen auch Symbole und Rituale eine Rolle, um Mitglieder persönlich an das Projekt zu binden. Die Stabilisierung auf der Ebene der einzelnen Projekte stärkt auch den Zusammenhalt des ganzen Feldes, das wiederum auf die Einzelnen zurückwirkt.

Kapitel 7

Theoretische Schlussbetrachtung

In den vorangegangenen Kapiteln wurden verschiedene strukturelle Merkmale des Feldes dargestellt, die auch teilweise schon in einen theoretischen Kontext gestellt wurden. Dies bezog sich sowohl auf einzelne Projekte als Teile des Feldes, wie auch auf das Feld insgesamt. Zudem wurden geschichtliche Hintergründe des Feldes aufgezeigt. Im folgenden soll nun versucht werden, das Feld als Ganzes theoretisch zu fassen. Dazu werden die in Kapitel 2 vorgestellten Modelle sozialer Formation herangezogen.

Wie gezeigt wurde gibt es ein hohes Maß an Heterogenität zwischen den Projekten. Gleichzeitig konnte ein hohes Maß an Homogenität unter den Entwicklern festgestellt werden, bezüglich ihrer sozialen Struktur wie auch ihres Selbstbildes. Deshalb erscheint es sinnvoll, die Ebene der Projekte und die Ebene des Feldes als Ganzes in der theoretischen Betrachtung zu trennen.

Das Feld als Ganzes ist durch eine rechtliche Grenze vom Rest der Welt abgetrennt. Die Lizenzen, die den Raum des Feldes definieren, sind im Urheberrecht verankert, also ist die Grenze von Faktoren außerhalb des Feldes abhängig. Das Überschreiten dieser Grenze, also das Mitgliedwerden, ist ein bewusster, rationaler Akt. Er setzt voraus, dass man die in den Lizenzen niedergelegten Prinzipien als Spielregeln des Feldes akzeptiert. Es handelt sich also um einen freiwilligen Prozess des Beitritts.

Die Betrachtung der Geschichte des Feldes und vor allem die Analyse des Selbstbildes der Entwickler hat gezeigt, dass es zudem eine starke kulturelle Übereinstimmung unter den Entwicklern herrscht, die sich in geteilten Werten und Normen niederschlägt. Der zentrale Wert ist die Auffassung, dass Information frei sein sollte, die auch in den Lizenzen verankert ist. Daher akzeptiert man mit den rechtlichen Grundlagen auch ein bestimmtes Wertesystem. Diese Werte sind für viele ein wichtiger Grund, sich als Entwickler zu beteiligen. Allerdings sind andere Motive noch wichtiger, die nicht ideologischer, sondern technisch-praktischer oder hedonistischer Natur sind.

Ein großer Teil der Werte und Normen bezieht sich nicht auf politische oder ideelle Fragen, sondern auf den Code selbst und die Form der Zusammenarbeit zwischen Entwicklern. Dies weist darauf hin, dass die Kultur der Entwickler Freier Software eher eine Arbeitskultur ist. Der Spaß am Verstehen technischer Vorgänge, ein ausgeprägter Forschungsdrang und das Beweisen der eigenen Fähigkeiten sind zentrale Merkmale der Kultur der Entwickler.[1] Diese Betonung des Wissens und der Programmierfähigkeiten im Selbstbild dient auch zur Abgrenzung gegen andere und ergänzt die rechtlich-ideologische Abgrenzung. All dies wird von den Entwicklern auch reflektiert und generalisiert, wodurch es zu einer Kanonisierung dieser Einstellungen kommt. Unterstützt wird dieser Prozess durch die Schaffung gemeinsamer Traditionen. Bestimmte Geschichten und Anekdoten werden immer wieder zur Illustration bestimmter Werte zitiert.

Die Programmierarbeit selbst ist also von besonderer Bedeutung. Sie findet in Projekten statt, die einzelne voneinander separate Produktionsorte im Gesamtraum des Feldes darstellen. Zwar gibt es auch Kommunikation zwischen den Projekten und einen gewissen Grad an personeller Überschneidung, aber die Entwicklungsarbeit leisten die Projekte jeweils (im wesentlichen) autonom – wobei sie jederzeit für neue Entwickler und Beiträge Außenstehender offen bleiben. Die innere Strukturierung der Projekte unterscheidet sich deutlich. Sie wird primär durch die Frage bestimmt, wie trotz der Freiwilligkeit der Teilnahme – und damit dem Fehlen von Zwangsmitteln – eine effektive Zielerreichung (das Entwickeln von Software) gewährleistet werden kann. Die Ausformung im einzelnen folgt also funktionalen Überlegungen. Schon durch die unterschiedlichen Arbeitsfelder ergibt sich so eine große Heterogenität der konkreten Organisationsformen. Ein weiterer Aspekt, der die Struktur beeinflusst, ist die Frage, wie für die nötigen Entscheidungen Legitimation und damit Folgebereitschaft hergestellt werden kann. Auch hier finden die Projekte verschiedene Lösungen. Daraus ergeben sich trotz ähnlicher Problemlage und gleichartiger Kultur verschiedene politische Systeme in den Projekten. Um die Arbeitsumgebung zu stabilisieren und die dafür erforderlichen Normen zu vermitteln, entwickeln die Projekte unterschiedliche Formen der Enkulturation. Schließlich bildet sich eine Identifikation mit den Projekten heraus, die einerseits zur Stabilität der Projekte beiträgt, andererseits die Mitglieder zum Dabeibleiben motiviert. Die Enkulturation und die sich bildende Identität wirken sich wiederum positiv auf die Kohäsion des gesamten Feldes aus.

Neben den Produktionsorten finden sich im Feld auch Akteure, die sich auf politische bzw. rechtliche Fragen spezialisiert haben. Auch wenn sie aufgrund ihrer Ziele und ihrer Kultur eindeutig zum Feld gehören, unterscheiden sie sich von den Entwicklungsprojekten dadurch, dass sie nicht der Produktion von Software, sondern der Reproduktion der rechtlichen Grenze dienen.

[1] Dabei spielt auch das Gefühl der Macht über die Technik eine nicht unerhebliche Rolle.

Das Feld zeichnet sich durch eine Homogenität der Kultur und eine Heterogenität der Struktur aus. Entsprechend gibt es eine klar erkennbare Grenze nach außen, aber im Inneren des Feldes kein eindeutig definiertes Zentrum. Zwar bewirkt die technische Bedeutung einiger Projekte, dass andere auf diese stärker angewiesen sind – eine kulturelle Hegemonialität folgt daraus nicht. Allenfalls sorgt die Wichtigkeit und die allgemeine Verwendung bestimmter Software dafür, dass ihre Programmierer besondere Bekanntheit erreichen und die Beteiligung an den entsprechenden Projekten eine besonders hohe Reputation im gesamten Feld mit sich bringt.

Zieht man die oben beschriebenen Modelle zur Analyse heran, so zeigt sich zunächst, dass die Methoden der Netzwerkanalyse für das Feld hilfreiche Erkenntnisse über die Strukturierung des Feldes liefern. Zur Dichte lässt sich feststellen, dass auf der Ebene des gesamten Feldes nur ein geringes Maß direkter Kontakte besteht. Die Virtualität des Raumes und die zu ihrer Überbrückung eingesetzten technischen Mittel führen zur Dominanz vermittelter Verbindungen in Form der one-to-many Kommunikation über Mailinglisten und Internetforen. Innerhalb der Projekte ist die Dichte deutlich höher. Es zeigen sich jedoch deutliche Unterschiede zwischen großen und kleinen Projekten. Hier ist anzumerken, dass die großen Projekte sich wiederum in Subprojekte aufteilen, in denen wieder eine höhere Dichte herrscht. Für ein Netzwerk mit so großer Reichweite ist die Homogenität der Beteiligten untypisch. Typisch dafür ist allerdings die zu beobachtende Clusterbildung – die Konzentration der Beziehungen in den einzelnen Projekten bzw. Subprojekten. Allgemein sind das Feld als Ganzes und die Projekte offen für andere, der Grad der Ausschließlichkeit ist also gering. Allerdings existieren innerhalb einiger Projekte innere Kreise, die stark nach außen abgeschottet sind. Eine soziale Kontrolle findet vor allem innerhalb der Projekte statt. Allerdings ist durch das fast völlige Fehlen von Zwangsmitteln ein hohes Maß an Selbstkontrolle durch Internalisierung der Normen nötig. Im Feld sind eher schwache Bindungen zu beobachten, deren Bandbreite primäre auf Arbeitsbeziehungen begrenzt ist. Das Feld kann also als loses Netzwerk, das wiederum dichtere Netzwerke (die Projekte) enthält, beschrieben werden. Es hat eine große Reichweite und weist nur wenige geschlossene Bereiche auf. Die Beziehungen haben eine geringe Bandbreite.

Aus dieser Analyse ergibt sich zunächst, dass die Modelle der sozialen Gruppe und der Organisation sinnvoll nur auf die Projekte bezogen werden können. Generell kann man jedes Projekt als soziale Gruppe beschreiben. Sie schließt sich zusammen, um ein bestimmtes Programm neu zu entwickeln oder zu verbessern. Sie bilden informelle Strukturen heraus, die vor allem auf einer Rollenverteilung innerhalb der Gruppe beruhen. Von zentraler Bedeutung sind insbesondere die Rolle des Projektleiters, sowie die weiterer Führungspersonen (Maintainer etc.). Diese Strukturen bleiben Gegenstand einer dauernden Aushandlung, in der sich Phasen des Wandels mit Phasen der Verfestigung abwechseln. Es lässt sich beobachten, dass die Ausdif-

ferenzierung der Strukturen immer den Erfordernissen des Arbeitsprozesses folgt. Meist bedeutet dies, dass jemand eine anfallende Aufgabe spontan übernimmt und diese sich später zu einer Position verfestigt. Für die Besetzung der so herausgebildeten Führungspositionen kommen unterschiedliche Mechanismen zum Einsatz. Durch die oben beschriebenen Prozesse der Enkulturation werden interne Normen und Werte vermittelt und so der Zusammenhalt gestärkt. Der Unterschied zu anderen Projekten wird teilweise durch Symbole unterstrichen, die zu einem Wir-Gefühl der Gruppe beitragen. Im wesentlichen beruht diese Identität aber auf dem *gemeinsamen* Entwickeln einer bestimmten Software.

Da zwischen den Entwicklern eine regelmäßige face-to-face Kommunikation nicht möglich ist, müssen sie sich, unabhängig von ihrer Größe, organisatorischer Mittel wie Mailinglisten, Webseiten oder Diskussionsforen bedienen. Auch wenn diese zunächst Charakteristika der formalisierten Kommunikation in Organisationen aufweisen, bedeuten sie noch nicht den Übergang eine Projektes von einer mehr oder weniger gefestigten Gruppenstruktur in eine formale Organisation. Während sich bei den meisten Projekten wegen der Notwendigkeit der Koordinierung des Arbeitsprozesses oder ihrer Größe Elemente von Organisation finden, habe nur wenige Projekte den Schritt zu einer formalisierten und kodifizierten Organisationsstruktur vollzogen. Die Häufigkeit solcher Organisations-Elemente macht es bisweilen schwierig, einzelne Projekte noch klar als Gruppen oder schon als Organisationen zu klassifizieren. Als Unterscheidungskriterium scheint die Frage geeignet, ob eine Trennung zwischen Person und Amt gemacht wird, oder ob es sich um Rollendifferenzierung im Sinne sozialer Gruppen handelt. Ersteres findet sich bei dem durch Wahl besetzten Amt des Projektleiters bei Debian, während etwa bei Eisfair die Struktur von der engen Verbindung von Persönlichkeit und Rolle geprägt ist. Dies lässt sich auf der unteren Ebene der Maintainer auch noch bei Debian finden. Daher ist selbst dieses stark formalisierte Projekt also keine voll ausgeprägte formale Organisation.

Das zentrale Element des Konzept der sozialen Welt ist die Annahme, dass sie sich um eine Kernaktivität bildet. Bei der Betrachtung der Entwickler Freier Software als soziale Welt wäre diese Kerntätigkeit das Entwickeln von Software – es wurde bereits die Zentralität des Programmierens erwähnt. Es geht allerdings nicht nur um das Entwickeln von Software als solcher, sondern diese muss bestimmte Bedingungen erfüllen. Zunächst einmal muss ihre Lizensierung der Definition von Richard Stallman von der Freiheit der Software entsprechen. Daneben gibt es weitere Sollbestimmungen bzw. Qualitätsvorstellungen bezüglich des Codes und der Kooperation unter den Beteiligten bei der Entwicklung. Zum einen soll der Code elegant sein, zum anderen soll die Zusammenarbeit dem von Eric Raymond beschriebene Basarmodell entsprechen. Hierbei geht es also um die Authentizität der Ausführung der Kernaktivität. Es lässt sich auch beobachten, dass je näher Entwickler an dieses Ideal heranreichen, desto mehr Ansehen erlangen sie und können auch mehr Einfluss

ausüben. Desweiteren sind sowohl das Selbstbild der Entwickler als auch die organisatorischen Strukturen durch das Programmieren bzw. durch dessen Erfordernisse geprägt. Zudem grenzen die Kernaktivität und die mit ihr verbundene Vorstellungen die soziale Welt der Entwickler Freier Software auch von anderen sozialen Welten ab, wie die der proprietären Softwareentwickler. Die Entstehung beruhte nicht auf einer Ausdifferenzierung innerhalb einer anderen, elterlichen Welt, sondern schon eher auf einer Abspaltung. Dies trifft aber auch nur bedingt zu. Zu Beginn hatte alle Software die Gestalt Freier Software oder auch von Public Domain bis Unternehmen dazu übergingen den Quellcode geheimzuhalten und ihre Programme mit restriktiven Lizenzen zu versehen. Freie Software war gegenüber proprietärer Software keine Abspaltung, sondern eher ein Beibehalten der vorherigen Gepflogenheiten, die nun explizit formuliert wurden. Von da an standen die soziale Welten der Freier und der proprietären Software in Konkurrenz. Das Aufkommen des Open Source-Begriffes könnte man mit dem Kampf um Legitimation dieser beiden sozialen Welten verstehen. Open Source ist letztlich eine Art Marketingkampange für Freie Software, die mit der technischen Effizienz des auf der Freiheit der Software angewiesene Basarmodells argumentiert.

Die Entwickler Freier Software lassen sich also als soziale Welt fassen. Allerdings lässt sich die Theorie der sozialen Welten auf (fast) alle sozialen Phänomene und Formationen anwenden, die sich um eine Tätigkeit herum bilden. Allerdings lässt das Konzept die Art der Zusammengehörigkeit, soziale Prozesse und kulturelle Aspekte außer Acht, die nicht direkt mit der Kernaktivität verbunden sind.

Die Idee der Gemeinschaft geht von einer nicht funktionalen, (quasi-)natürlichen Zusammengehörigkeit aus. Es lässt sich im Feld aber eine starke Zweckhaftigkeit der Beziehungen identifizieren. Der Zusammenschluss der Entwickler begründet sich im gemeinsamen Schaffen von Software. Desweiteren besteht weder eine natürliche Verbindung zwischen den Entwicklern noch findet derzeit ein Prozess innerhalb des Feldes statt, eine solche zu konstruieren. Es sind allerdings Prozesse zu beobachten, die denen einer Vergemeinschaftung ähneln. So lassen sich eine endogene Geschichtsschreibung und eine Berufung auf gemeinsame Traditionen finden. Dieser werden aber bisher zur Erklärung kultureller Phänomene herangezogen und nicht zur bewussten Begründung von Gemeinsamkeit. Um die Entwickler Freier Software als Gemeinschaft fassen zu können, müsste zudem das Konzept der Gemeinschaft an auf virtuellen Beziehungen basierende soziale Formationen angepasst werden. Hierbei müsste allerdings darauf geachtet werden, dass das Konzept nicht soweit ausgeweitet wird, dass es dem der sozialen Gruppe entspricht und so seine analytische Schärfe verliert (vgl. Heintz 2003: 203f.). Ausgangspunkt für eine Betrachtung des Feldes als Gemeinschaft könnte Tönnies Konzept der Freundschaft sein, weil hier nicht-biologische Beziehungen die Grundlage bilden. Diese beruht stattdessen auf dem „bloßen Miteinander-Wirken und Walten in der gleichen Richtung, in glei-

chem Sinne“ und ist eine Gemeinschaft des Geistes (Tönnies 1991: 12[1. Buch § 6]). Zur Lösung der Problematik der Ortlosigkeit bzw. Virtualität könnten eventuell Theorien der Translokalität und der Migrationsforschung herangezogen werden.

Während die Theorie der sozialen Welt vor allem das gemeinsame Handeln betont, und das Konzept der Gemeinschaft die Qualität bzw. den Ursprung der Zusammengehörigkeit in den Mittelpunkt stellt, konzentriert sich die Theorie der Kultur auf die Ausgestaltung der Integration des Zusammenschlusses und die Übereinstimmung des Weltbildes der Mitglieder. Kultur ist dabei zu verstehen als geteilter Wahrnehmungs- und Bewertungsrahmen, der das gemeinsame Leben und Handeln regelt. Dieser geteilte Rahmen schlägt sich in konkreten Werten und Normen nieder. Das Feld beinhaltet nur einen Ausschnitt des Lebens der beteiligten Entwickler. Will man es mit der Theorie der Kultur betrachten, muss also zunächst die Einschränkung gemacht werden, dass es sich dabei nur um eine Subkultur handeln kann. Eine solche unterscheidet sich von einer Gesamtkultur dadurch, dass sie nur einzelne Lebensbereiche regelt und ein entsprechend darauf beschränktes Set von Werten und Normen hervorbringt. Eine Subkultur ist immer bezogen auf eine dominante Mehrheitskultur. Sie weicht von dieser bezüglich einiger weniger Weltdeutungen ab. Für bestimmte Aspekte der Realität findet sie also alternative Erklärungen, auf die sich ein kulturelles Instrumentarium aufbaut. Bezogen auf das Feld Freier Software lassen sich hier große Übereinstimmungen finden. Zum einen wird durch die starke Betonung des Zwecks des Zusammenschlusses (gemeinsam Software zu entwickeln) durch die Beteiligten deutlich, dass es hier nur um einen Teil ihres Lebens geht. Gleichzeitig spielen aber ideologische Motive eine wichtige Rolle. Die Entwickler teilen in einem hohem Maße die Ansicht, dass Software frei sein sollte, was eine deutliche Abweichung von Vorstellungen über geistiges Eigentum in der Mehrheitskultur darstellt. Dies unterscheidet sich auch deutlich vom reinen Ignorieren der Regelungen der Mehrheitsgesellschaft bezüglich solchen Eigentums, also vom Raubkopieren. Es geht also nicht nur darum, Software zu besitzen, sondern es handelt sich um eine anderes Verständnis vom Besitz von Software. Dies stellt eine inhaltliche Abweichung von der Mehrheitskultur dar, jedoch kein gezieltes Arbeiten gegen sie – es ist also eine Sub-, keine Kontrakultur. Während die hohe Bedeutung des individuellen Besitzes zurücktritt, werden bestimmte Vorstellungen von Qualität wichtiger. Diese grundlegenden Ansichten werden in entsprechende Werte und Normen umgesetzt, die sich auf den Code, dessen Freiheit, die Zusammenarbeit und den Stellenwert von Wissen beziehen. Diese Werte und Normen werden wie die Kultur insgesamt durch bestimmte Mechanismen reproduziert. Die Lizenzen kodifizieren das grundlegende Verständnis von Freiheit und das Copyleft sichert dieses dauerhaft ab. Auch die Arbeitsstrukturen dienen der Befestigung und Umsetzung der Werte und Normen. Durch ihre Offenheit und flexible Anpassung an die konkrete Aufgabe setzen sie die Zentralität der Qualität um. Die Projekte, die enger verbundene Gruppen

sind, übernehmen einen großen Teil der Enkulturation – häufig durch strukturierte Prozesse – und auch der Kontrolle der Einhaltung der Werte und Normen. Dies wird ergänzt durch eine feldweite Kulturproduktion, die ihre eigenen Mythen, Traditionen, Bräuche und Symbole hervorbringt. Trotz der hohen Übereinstimmung der empirischen Erkenntnisse mit der Theorie der Subkultur, weist dieser Ansatz ein Problem auf – die Wichtigkeit der Tätigkeit des Entwickelns Freier Software wird nur unzureichend berücksichtigt. Eine weitere Frage ergibt sich daraus, dass die Entwickler zwar eine gemeinsame Kultur teilen, solang sie sich innerhalb des Feldes bewegen – der von dieser Subkultur nicht erfasste Rest befindet sich jedoch in verschiedenen Mehrheitskulturen, da die Entwickler über die ganze Welt verteilt sind.

In Beschreibungen des Feldes ist häufig von einer Bewegung Freie Software die Rede. Auf den ersten Blick lassen sich auch durchaus Belege für diese Wahrnehmung finden. Die alternative Lizenzpolitik, die in der Öffentlichkeit auch im Zusammenhang mit der Kritik an Softwarepatenten etc. wahrgenommen wird, scheint ein Beispiel dafür zu sein, dass ein bestimmter politischer bzw. wirtschaftlicher Trend – die Monopolbildung – im Stil einer Bewegung abgewehrt werden soll. Positiv kann das Eintreten für Freie Software auch als Hinarbeiten auf eine gesellschaftliche Veränderung verstanden werden. Die heterogene Netzwerkstruktur scheint das ebenso zu unterstreichen wie die enge Verbindung von Personen und Anliegen. Diese Analyse übersieht jedoch wesentliche Aspekte sowohl des Feldes wie auch von sozialen Bewegungen (vgl. z.B. Zimmermann 2004). Wie die Betrachtung des Feldes gezeigt hat, steht bei den Entwicklern Freier Software insgesamt nicht die gesellschaftliche Veränderung im Vordergrund, sondern vielmehr die Herstellung von Software. Die ideologischen Fragen haben zwar eine Bedeutung, sie sind aber nicht der Grund des Handelns. Und die wesentliche Tätigkeit ist nicht das Umsetzen der ideologischen Ansichten, diese dienen vielmehr als Grundlage der Kultur und als Basis für das Handeln. Der hohe Anteil von Entwicklern, der beruflich mit dem Entwickeln proprietärer Software beschäftigt ist, erschwert eine ideologische Erklärung des Handelns. Natürlich lassen sich (etwa im Umfeld des GNU-Projekts) auch Entwickler finden, für die ideologische Motive von zentraler Bedeutung sind und die ihre Arbeit vor allem als politisch motiviert verstehen. Aber selbst diese haben ihre politische Aktivität in dafür geschaffene besondere Organisationen verlegt (in diesem Fall die FSF). Diese Organisationen beschäftigen sich nicht (mehr) mit dem Entwickeln von Programmen, sondern ausschließlich mit rechtlichen Fragen und politischer Lobbyarbeit. Man kann also durchaus sagen, dass ein Teil des Feldes Bewegungscharakter hat. Allerdings nur in der Form, dass es darin politische Organisationen gibt, die unter den restlichen Mitgliedern des Feldes mobilisierbare Anhänger finden. Das Gros der kollektiven Akteure im Feld – also die Projekte – hat jedoch kein politisches Anliegen als Ziel. Umgekehrt ist es eher so, dass Teile des Feldes Freier Software, die

politischen Institutionen wie FSF und OSI, Teil einer größerer sozialen Bewegung sind, zu dem auch die EFF, die FFII und andere gehören. Diese Bewegung hat die Aufrechterhaltung bürgerlicher Freiheiten in einer kommerzialisierten Wissensgesellschaft zum Thema.

Kapitel 8

Fazit und Ausblick

Zu Beginn der Arbeit wurde das Feld Freie Software als für die Ethnologie ein wenig untypisch bezeichnet – jedenfalls nach einem klassischen Begriff des Fachs. Im Laufe der Analyse hat sich aber gezeigt, dass sich im Feld zahlreiche Phänomene finden lassen, die auch in der herkömmlichen außereuropäischen Feldstudie Thema sind. Dazu gehören etwa politische Systeme wie Häuptlingstümer und akephale Ordnungen, die nicht kodifiziert sind. Entsprechend wichtig sind für diese Ordnungen die kulturelle Verankerung und integrative Mechanismen, die sie trotz einer relativ ausgeprägten strukturellen Flexibilität dauerhaft stabilisieren.

Es handelt sich bei dem Feld eindeutig um ein Produktions- und Austauschsystem, das aber ohne Geld auskommt. Es hat reziproken Charakter, der Austausch ist nicht marktförmig organisiert. Entsprechend spielen monetärer Gewinn und Profitmaximierung keine Rolle. Diese Form des Austausches von für das Feld lebenswichtigen Gütern – in diesem Fall: Wissen und Code – ähnelt sehr stark den aus der klassischen Ethnologie bekannten Ökonomien. Ein großer Unterschied besteht darin, dass der Geber das Gut gleichzeitig selbst behalten kann. Die weitere Untersuchung dieses letzten Aspektes mit den Methoden der Wirtschaftsethnologie ist für die Bildung einer Theorie der Wissensökonomie von hohem Interesse. Umgekehrt würde eine solche Untersuchung dazu beitragen, für die Ethnologie neue Perspektiven auf eine bekannte Thematik zu eröffnen.

Auch der Blickwinkel einer Ethnologie der Arbeit könnte hilfreich sein, um neue Erkenntnisse über verteilte Arbeitsprozesse in virtuellen sozialen Formationen zu gewinnen. Das Studium der Softwareentwicklung und damit der Produktion eines Gutes, das auf angewandtem Wissen beruht und gleichzeitig selbst wieder Wissenscharakter hat, könnte zudem neue theoretische Erkenntnisse über Arbeitsprozesse erbringen.

Kapitel 9

Literatur

Allner, Jörg/Allner, Kerstin (2003): Computer Classics. Düsseldorf: Data Becker.

Anderson, Benedict (1983): Imagined Communities: Reflections on the Origin and Spread of Nationalism. London: Verso.

Babcock, Charles (2001): MS Attacks Open Source.
http://www.eweek.com/article2/0,1759,1243775,00.asp am 18.7.2004.

Baumgärtel, Tilman (2002): Am Anfang war alle Software frei. Microsoft, linux und die Rache der Hacker. In: Roesler, Alexander/Stiegler, Bernd (Hgs.): Microsoft. Medien – Macht – Monopol. Frankfurt/Main: Suhrkamp, S. 103-129.

Becker, Konrad u.a. (2002): Die Politik der Infospähre. World-Information.Org. Bonn: Bundeszentrale für politische Bildung.

Berners-Lee, Tim (1999): Der Web-Report. München: Econ Verlag.

Bochers, Detlef/Kuri, Jürgen (2003): SCO vs. Linux: Eingebettet ruht sich's sanft. Heise Online News vom 27.8.2003.
http://www.heise.de/newsticker/meldung/39793 am 21.9.2004.

Brand, Andreas (o. J.): Fallstudie horizontales elektronisches Arbeitsnetzwerk/Open Source-Projekt. Unveröffentlichtes Manuskript.

Bushell, Thomas (1996): Hurd 0.0 and GNU 0.0 released.
http://www.gnu.org/software/hurd/hurd-flash13 am 26.9.2004.

Calhoun, Craig (1991): Indirect Relationships and Imagined Communities: Large-Scale Social Integration and the Transformation of Everyday Life. In Bourdieu, P. /Coleman J.S. (Hg.): Social Theory for a Changing Society. Boulder, CO: Westview Press and New York: Russell Sage Foundation, S. 95-121.

Castells, Manuell (2001): Der Aufstieg der Netzgesellschaft. Opladen: Leske + Budrich.

Ceruzzi, Paul E. (2003): Eine kleine Geschichte der EDV. Bonn: Mitp-Verlag.

Chandar, Anupam (2003): This Penguin my bite. Linux's counterattack against SCO.
http://writ.news.findlaw.com/commentary/20031113_chander.html
am 15.11.2003.

Claessens, Dieter (1983): Die Gruppe unter innerem und äußerem Organisationsdruck. In: König, Rene/Neidhardt, Friedhelm/Lepsius, M. Rainer (Hg.): Gruppensoziologie. Perspektiven und Materialien. Opladen: Westdeutscher Verlag, S. 484-496.

DARPA (2003): ARPA-DARPA: The History of the Name.
http://www.darpa.mil/body/arpa_darpa.html am 15.7.2004.

Debian (2003): Debian New Maintainers' Corner.
http://www.debian.org/devel/join/newmaint am 2.7.2003.

Delio, Michelle (2001): Who's Better: Geeks or Nerds?. Wired News vom 2. 2. 2001.
http://www.wired.com/news/technology/0,1282,41422,00.html am 21.9.2004.

DiBona, Chris/Ockman, Sam/Stone, Mark (1999): Introduction. In: DiBona, Chris / Ockman, Sam / Stone, Mark: Open Source. Voices from the Open Source Revolution. Sebastopol: O'Reilly, s. 1-17.

Elwert, Georg (1980): Die Elemente der traditionellen Solidarität. eine Fallstudie in Westafrika. In: Kölner Zeitschrift für Soziologie und Sozialpsychologie Jg. 32, S. 681-704.

Elwert, Georg (1989): Nationalismus und Ethnizität. Über die Bildung von Wir-Gruppen. In: Kölner Zeitschrift für Soziologie und Sozialpsychologie, Jg. 41, S. 440-464.

Elwert, Georg (1999): Eigentum. In: Betz, Hans Dieter et al.: Religion in Geschichte und Gegenwart. Bd. 2. Tübingen: Mohr, Siebeck, S. 1143.

Elwert, Georg (2001): Sozialanthropologischen und ethnologischen Gewaltforschung. Berlin: unveröffentlichtes Manuskript.

Ettrich, Matthias (2004): Koordination und Kommunikation in Open Source Projekten. In: Gehring, Robert A./Lutterbeck, Bernd (Hg.): Open Source Jahrbuch 2004. Berlin: Lehmanns Media, S. 179-192.

Farlex (2004): WorldForge. In: TheFreeDictionary.com.
http://encyclopedia.thefreedictionary.com/WorldForge am 30.9.2004.

fli4l (o. J.): Was ist Fli4L?. http://www.fli4l.de/german/fli4l.htm am 29.9.2004.

Free Software Foundation (2002a): GNU Lesser General Public License, Version 2.1, February 1999. http://www.fsf.org/licenses/lgpl.html am 24.10.2003.

Free Software Foundation (2002b): Categories of Free and Non-Free Software. http://www.gnu.org/philosophy/categories.html am 13.9.2004.

Free Software Foundation (2003a): GNU General Public Licence, Version 2, June 1991. http://www.fsf.org/licenses/gpl.html am 24.10.2003.

Free Software Foundation (2003b): Frequently Asked Questions about the GNU GPL. http://www.fsf.org/licenses/gpl-faq.html am 24.10.2003.

Free Software Foundation (2004a): Some Confusing or Loaded Words and Phrases that are Worth Avoiding.
http://www.gnu.org/philosophy/words-to-avoid.html am 25.9.2004.

Free Software Foundation (2004b): Why „Free Software" is better than „Open Source".
http://www.gnu.org/philosophy/free-software-for-freedom.html am 25.9.2004.

Freiberger, Paul/Swaine, Michael (2000): Fire in the Valley. The Making of the Personal Computer. New York: McGraw Hill.

Gates, Bill (1976): Open letter to Hobbyists.
http://www.blinkenlights.com/classiccmp/gateswhine.html am 13.9.2004.

Gennep, Arnold van (1999): Übergangsriten. Frankfurt/Main. Campus.

Gentoo Foundation (2001-2004): Gentoo Linux Social Contract.
http://www.gentoo.org/main/en/contract.xml am 19.8.2004.

Ghosh, Rishab Aiyer/Glott, Ruediger/Krieger, Bernhard/Robles, Gregorio (2002): Free/Libre and Open Source Software: Survey and Study. Part 4: Survey of Developers.
http://www.infonomics.nl/FLOSS/report/FLOSS_Final4.pdf am 14.9.2004.

Girtler, Roland (1991): Forschung in Subkulturen. In: Flich, Uwe (Hg.): Handbuch qualitative Sozialforschung. Grundlagen, Konzepte, Methoden und Anwendungen. München: Psychologie-Verl.-Union, S. 385-388.

Girtler; Roland (1995): Randkulturen. Theorie der Unanständigkeit. Wien; Köln; Weimar: Böhlau.

Glass, Adam (1994): Theo de Raadt.
http://mail-index.netbsd.org/netbsd-users/1994/12/23/0000.html am 5.10.2004.

Gooch, Richard (2004): The linux-kernel mailing list FAQ.
http://www.kernel.org/pub/linux/docs/lkml/ am 28.9.2004.

Granovetter, Mark (1973): The strength of weak ties. In American Journal of sociology, 78, S. 1360-1380.

Grassmuck, Volker (2002): Freie Software. Zwischen Privat- und Gemeineigentum. Bonn: Bundeszentrale für politische Bildung.

Greve, Georg (2004): Brave GNU World. In: Linux Magazin, Jg. 10, Nr. 4, S. 102-105.

Gukenbiehl, Hermann L. (2002): Institution und Organisation. In: Korte, Hermann/
Schäfers, Bernhard: Einführung in Hauptbegriffe der Soziologie. Opladen: Leske + Budrich. 6. Aufl..

Hagen, Wolfgang (2002): Bill Luhan und Marshall McGates. Die Extension des Menschen als Extension der USA. In: Roesler, Alexander/Stiegler, Bernd (Hgs.): Microsoft. Medien – Macht – Monopol. Frankfurt/Main: Suhrkamp, S. 24-47.

Hann, C. M. (1998): Introdution: the embeddedness of property. In Hann, C. M.: Property Relations - Renewing the anthropological tradition. Cambridge, Cambridge UP, S. 1-47.

Heintz, Bettina (2003): Gemeinschaft ohne Nähe? Virtuelle Gruppen und reale Netze. In: Thiedeke, Udo: Virtuelle Gruppen. Charakteristika und Problemdimensionen. Wiesbaden: Westdeutscher Verlag, 2. Aufl., S. 180-210.

Helmers, Sabine (1994): Internet im Auge der Ethnologin.
http://duplox.wz-berlin.de/texte/ding/index.html am 10.9.2004.

Hillmann, Karl-Heinz (1994): Wörterbuch der Soziologie. Stuttgart: Kröner. 4. Auflage

Hiltzik, Michael A.(2000): Dealers of Lightning. Xerox Parc and the Dawn of the Computer Age. New York, NY: HarperCollins.

Hirschman, Albert O. (174): Abwanderung und Widerspruch. Reaktionen auf Leistungsabfall bei Unternehmen, Organisationen und Staaten. Tübingen: J. C. Mohr (Paul Siebeck).

The Jargon Dictionary (2000a): Cracker.
http://info.astrian.net/jargon/terms/c/cracker.html am 21.7.2004.

The Jargon Dictionary (2000b): Warez dOOdz.
http://info.astrian.net/jargon/terms/w/warez_d00dz.html am 21.7.2004.

Kaplan, David A. (2000): Silicon Valley. Die digitale Traumfabrik und ihre Helden. München: Heyne.

KDE (o. J.): KDE e.V. Homepage. What is K Desktop Environment e.V.?
http://www.kde.org/areas/kde-ev/ am 29.9.2004.

Kofler, Michael (2001): Linux. Installation, Konfiguration, Anwendung. München: Addison-Wesley, 6. Aufl..

Kohl, Karl-Heinz (1993): Ethnologie - die Wissenschaft vom kulturellen Fremden. Eine Einführung. München: C. H. Beck Verlag.

Kollock, Peter/Smith Marc A. (1999): Communities in cyperspace: In: Kollock, Peter/Smith Marc A. (Hg.): Communities in Cyperspace. London: Routledge, S. 3-25.

Kreuzer, Till (2004): Gründe für GPL-Urteil veröffentlicht (26.07.2004). http://www.ifross.org/ifross_html/home2_2004.html am 6.10.2004.

Kuri, Jürgen (2004): SCO vs. Linux: Die unendliche Geschichte. c't aktuell vom 10.02.2004. http://www.heise.de/ct/aktuell/meldung/44492 am 1.8.2004.

Lakhani, Karim R./Wolf, Robert G. (2003): Why Hackers Do What They Do? Understanding Motivation and Effort in Free/Open Source Software Projcts. http://opensource.mit.edu/online_papers.php am 7.10.2004.

Lerner, Josh/Tirole, Jean (2000): The simple Economics of Open Source. http://opensource.mit.edu/online_papers.php am 7.10.2004.

Lessig, Lawrence (2001): Code und andere Gesetze des Cyberspace. Berlin: Berlin Verlag.

Levy, Steven (2001): Hackers. Heros of the Computer Revolution. London: Penguin Books.

Linux International (1994-2004): Jon "maddog" Hall. http://www.li.org/who/bio.php?name=hall am 8.9.2004.

LinuxWorld Expo (2004): Keynotes & Feature Presentations. http://www.linuxworldexpo.com/linuxworldny/V40/index.... am 22.1.2004.

Luthiger, Benno (2004): Alles aus Spaß? zur Motivation von Open Source Entwicklern. In: Gehring, Robert A./Lutterbeck, Bernd (Hg.): Open Source Jahrbuch 2004. Berlin: Lehmanns Media, S. 93-106.

Lyx-Team (2002): Das LyX-Tutorium. Lyx Version 1.3.2 In: Suse: Linux Professional 9.0.

Mandrakesoft (o. J.): Über uns. http://www.mandrakesoft.com/company/about am 25.7.2004.

Marx, Karl (1974): Das Kapital. Kritik der ökonomischen Theorie. 1. Bd. Berlin: Dietz Verlag.

Mauss, Marcel (1990): Die Gabe. Form und Funktion des Austauschs in archaischen Gesellschaften. Frankfurt/Main: Suhrkamp.

McKusick, Marshall Kirk (1999): Twenty Years of Berkeley Unix: From AT&T-Owned to Free-Redistributable. In: DiBona, Chris / Ockman, Sam / Stone, Mark: Open Source. Voices from the Open Source Revolution. Sebastopol: O'Reilly, S. 31-46.

Meyer, Frank (2002): eisfair - Easy Internet Server V1.0.1. http://eisfair.org/german/eisfair.htm am 29.9.2004.

Michlmayr, Martin (2003): Managing Debian. Konferenz-DVD Linuxtag 2004.

Microsoft (1995): Word 1997. Endnutzer-Lizenzvertrag für Mircosoft-Software. unveröffentlicht.

Moody, Glyn (2001): Die Software Rebellen. Die Erfolgsstory von Linus Torvalds und Linux. Landsberg/Lech: Verlag Moderne Industrie.

Murdock, Ian (2003): Debian: A Brief Retrospective. http://www.linuxplanet.com/linuxplanet/editorials/4959/1/ am18.8.2004.

Neidhardt, Friedhelm (1983): Themen und Thesen zur Gruppensoziologie. In: König, Rene/Neidhardt, Friedhelm/Lepsius, M. Rainer (Hg.): Gruppensoziologie. Perspektiven und Materialien. Opladen: Westdeutscher Verlag, S. 12-34.

NetBSD Foundation (2004): The History of the NetBSD Project. http://www.netbsd.org/Misc/history.html am 16.7.2004.

Nuss, Sabine (2002): Download ist Diebstahl? Eigentum in einer digitalen Welt. In: PROKLA. Zeitschrift für kritische Sozialwissenschaft, jg. 32, Heft 126, S. 11-35.

o. V. (1999): The Tannenbaum-Torvalds Debate. In: DiBona, Chris / Ockman, Sam / Stone, Mark: Open Source. Voices from the Open Source Revolution. Sebastopol: O'Reilly, S. 221-251.

Open Source Technology Group (2004): freshmeat.net: Browse project tree – licene :: OSI approved. http://freshmeat.net/browse/14/ am 18.82004.

Oringel, Amy (1998): The bazaar finds its missionary in Eric Raymond. Developers.com – The Journal am 20.5.1998. http://www.developer.com/journal/profiles/052098_raymond.html am 10.7.2000.

Osterloh, Margit/Rota, Sandra/Kuster, Bernhard (2004): Open-Source-Software: Ein neues Innovationsmodell? In: Gehring, Robert A./Lutterbeck, Bernd (Hg.): Open Source Jahrbuch 2004. Berlin: Lehmanns Media, S. 121-137.

Peuckert, Rüdiger (2002): Abweichendes Verhalten und soziale Kontrolle. In: Korte, Hermann/Schäfers, Bernhard: Einführung in Hauptbegriffe der Soziologie. Opladen: Leske + Budrich. 6. Aufl..

Raymond, Eric (1999a): A Brief History of Hackerdom. In: DiBona, Chris / Ockman, Sam / Stone, Mark: Open Source. Voices from the Open Source Revolution. Sebastopol, CA: O'Reilly, S. 19-29.

Raymond, Eric (1999b): The Revenge of the Hackers. In: DiBona, Chris / Ockman, Sam / Stone, Mark: Open Source. Voices from the Open Source Revolution. Sebastopol, CA: O'Reilly, S. 207-219.

Raymond, Eric (2000a): The Cathedral and the Bazaar. Version 3.0. http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/ am 1.8.2004.

Raymond, Eric (2000b): Homesteading the Noosphere. Version 3.0. http://www.catb.org/~esr/writings/cathedral-bazaar/homesteading am 1.8.2004.

Raymond, Eric (2000c): The Magic Cauldron. Version 3.0 http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/ am 6.9.2004.

Raymond, Eric (Hg.) (2001): The New Hacker's Dictionary. VERSION 4.3.1, 29.6.2001. http://www.iwar.org.uk/hackers/resources/faq/jargon.htm am 15.9.2004.

Raymond, Eric (2004): Eric's Gun Nut Page. http://www.catb.org/~esr/guns/ am 7.9.2004.

Regents of the University of California (1993): BSD Licence. http://www.xfree86.org/3.3.&/copyright2/html#5 am 26.7.2004.

Richter, Rudolf/Furubotn, Eirik G. (1996): Neue Institutionenökonomik. Tübingen: Mohr.

Rucht, Dieter/Neidhardt, Friedhelm (2001): Soziale Bewegungen und kollektive Aktionen. In: Joas, Hans: Lehrbuch der Soziologie. Frankfurt/Main: Campus. S. 533-556

Sack, Fritz (1971): Die Idee der Subkultur. Eine Berührung zwischen Anthropologie und Soziologie. In: Kölner Zeitschrift für Soziologie und Sozialpsychologie, Jg. 23, S. 261-282.

Salus, Peter H. (1995): A Quater Century of Unix. Reading, MA: Addison-Wesley Publishing Company.

Schimank, Uwe (2001): Gruppen und Organisationen. In: Joas, Hans: Lehrbuch der Soziologie. Frankfurt/Main: Campus. S. 199-222

Shankland, Stephen (2001): Microsoft license spruns open source. http://news.com.com/2100-1001-268889.html?legacy=cnet am 18.7.2004.

Shimizu, Hiroyuki/Ilo, Jun/Hiyane, Kazuo (2004): The whole image of free/ libre/ open source software developers in Japan and Asia. Unveröffentlichtes Manuskript.

Short, Chris/Short, Carri (2003): History of Linux. http://www.shortfamilyonline.com/tech/unix/history-of-linux/ am 25.7.2004.

Spittler, Gerd (1980): Streitreglung im Schatten des Leviathan. eine Darstellung und Kritik rechsethnologischer Untersuchungen. In: Zeitschrift für Rechtssoziologie, Jg. 1, Heft 1, S. 4-32.

Stallman, Richard (1999): The GNU Operating System and the Free Software Movement. In: DiBona, Chris / Ockman, Sam / Stone, Mark: Open Source. Voices from the Open Source Revolution. Sebastopol: O'Reilly, S. 53-70.

Stallman Richard (2004): The Hurd an Linux.
http://www.gnu.org/software/hurd/hurd-and-linux.html am 26.9.2004.

Strauss, Anselm (1978): A Social World Perspective. In: Studies in Symbolic Interaction. Vol. 1, S. 119-128.

Strauss, Anselm (1982): Social Worlds and Legitimation Processes. In: Studies in Symbolic Interaction. Vol. 4, S. 171-190.

Strauss, Anselm (1984): Social Worlds and their Segmentation Processes. In: Studies in Symbolic Interaction. Vol. 5, S. 123-139.

Streinz, Rudolf (2001): Europarecht. 5. Aufl. Heidelberg: C.F. Müller.

Strübing, Jörg (2002): Pragmatische Heuristik. Interaktionistische Beiträge zur Erforschung von Wissenschaft und Technik als Arbeit. Habilitationsschrift.

Suse (2003): SUSE LINUX - A Novell Company. Milestones in Technology Leadership.
http://www.suse.de/de/company/suse/suse/milestones.html am 25.7.2004.

Tatham, Simon (1999): How to Report Bugs Effectively.
http://www.chiark.greenend.org.uk/~sgtatham/bugs.html am 14.9.2004

Tönnies, Ferdinand (1991): Gemeinschaft und Gesellschaft. Darmstadt: Wissenschaftliche Buchgesellschaft. 3. Aufl..

Torvalds, Linus (1991): What would you like to see most in minix?
http://groups.google.at/groups?selm=1991Oct5.071651.9658%40agate.berkeley.edu am 24.7.2004.

Torvalds, Linus/Diamond, David (2001a): Just For Fun. Wie ein Freak die Computerwelt revolutionierte. München: Carl Hanser Verlag.

Torvalds, Linus (2001b): Was geht in Hackern vor? Oder: Das Linussche Gesetz. In Himanen, Pekka: Die Hacker-Ethik und der Geist des Informations-Zeitalters. München: Riemann Verlag. S. 13-18.

Trolltech (o.J.): Qt Free Edition Licensing.
http://www.trolltech.com/products/qt/freelicense.html am 29.9.2004.

von Trotha, Trutz (2000): Was ist Recht? Von der gewalttätigen Selbsthilfe zur staatlichen Rechtsordnung. In: Zeitschrift für Rechtssoziologie, Jg. 21, Heft 2, S. 325-354.

Turbolinux (o. J.): About us. http://www.turbolinux.com/about/ am 25.7.2004.

Vaughn-Nichols, Steven J. (2003): Linus Torvalds Refutes SCO Copyright Claims. In: Week Enterprise News and Reviews am 22.12.2003.
http://www.eweek.com/print_article/0,3048,a=115229,00.sap am 18.1.2004.

Weber, Max (1978): Methodologische Schriften. Frankfurt/Main: Fischer.

Weber, Max (1980): Wirtschaft und Gesellschaft. Grundriss der verstehenden Soziologie. Tübingen. J. C. B. Mohr. 5. Aufl.

Weizenbaum, Joseph (1978): Die Macht der Computer und die Ohnmacht der Vernunft. Frankfurt/Main: Suhrkamp.

Wellman, Barry (2003): Die elektronische Gruppe als soziales Netzwerk. In Thiedeke, Udo: Virtuelle Gruppen. Wiesbaden: Westdeutscher Verlag, 2. Aufl.. S. 126-159.

Williams, Sam (2002): Free as in Freedom. Richard Stallman's Crusade for Free Software. Sebastopol, CA: O'Reilly.

Wolfinger, Raymond E. (1960): Reputation and Reality in the Study of „Community Power". In: American Sociological Review. Vol. 25, No. 5., S. 636-644.

Yinger, J. Milton (1960): Contraculture and Subculture. In: American Sociological Review. Vol. 25, No. 5., S. 625-635.

Young, Jeffrey S. (1989): Steve Jobs. Der Henry Ford der Computerindustrie. Düsseldorf: Systemtechnik.

Young, Robert/Goldman Rohm, Wendy (2000): Der Red Hat Coup. Bonn: MITP-Verlag.

Zimmermann, Thomas (2004): Open Source und Freie Software - soziale Bewegung im virtuellen Raum? In: Gehring, Robert A./Lutterbeck, Bernd (Hg.): Open Source Jahrbuch 2004. Berlin: Lehmanns Media, S. 121-137.

Anhang A

Tabellen

Tabelle 1: Geschlechterverteilung

	Anzahl	**%**	**Valide %**
Mnnlich	213	97,5	97,7
Weiblich	5	2,2	2,3
Missing	218	2,2	

Tabelle 2: Altersverteilung

	Anzahl	**%**	**Valide %**
Mnnlich	213	97,5	97,7
Weiblich	5	2,2	2,3
Missing	218	2,2	

Tabelle 3: Feste Partnerschaft

	Anzahl	**%**	**Valide %**
Yes	109	48,9	50
No	109	48,9	50
Missing	5	2,2	

Tabelle 4: Kinder

	Anzahl	**%**	**Valide %**
Yes	32	14,3	14,5
No	188	84,3	85,3
Missing	3	1,3	

Tabelle 5: Herkunft

	Anzahl	%	Valide %
Antigua and Barbuda	1	0,4	0,5
Argentina	1	0,4	0,5
Australia	10	4,5	4,5
Austria	2	0,9	0,9
Bahamas	1	0,4	0,5
Belgium	5	2,2	2,3
Brazil	1	0,4	0,5
Canada	10	4,5	4,5
Colombia	1	0,4	0,5
Czech Republic	3	1,3	1,4
Denmark	2	0,9	0,9
Estonia	1	0,4	0,5
Finland	3	1,3	1,4
France	17	7,6	7,7
Germany	51	22,9	23,2
Greece	2	0,9	0,9
India	2	0,9	0,9
Ireland	1	0,4	0,5
Israel	7	3,1	3,2
Italy	4	1,8	1,8
Jamaica	1	0,4	0,5
Japan	1	0,4	0,5
Netherlands	9	4	4,1
Norway	3	1,3	1,4
Poland	2	0,9	0,9
Portugal	2	0,9	0,9
Romania	1	0,4	0,5
Russian Federation	2	0,9	0,9
Slovenia	1	0,4	0,5
South Africa	2	0,9	0,9
Spain	2	0,9	0,9
Sweden	7	3,1	3,2
Switzerland	3	1,3	1,4
Turkey	1	0,4	0,5
Great Britain	13	5,8	5,9
United States of America	45	20,2	20,5
Missing	3	1,3	

Tabelle 6: Höhster erreichter Bildungsabschluss

	Anzahl	%	Valide %
Primary school	2	0,9	0,9
Secundary School	13	5,8	5,9
A-level	65	29,1	29,3
Bachelor	72	32,3	32,4
Masters	57	25,6	25,7
Doctorate	12	5,4	5,4
Apprenticeship/Occupational training	1	0,4	0,5
Missing	1	0,4	

Tabelle 7: Einstellung zu proprietärer Software

	Anzahl	%	Valide %
Proprietary software is not acceptable. FS/OS is the only way	21	9,4	9,5
FS/OS is preferable but people might have their reasons to use properietray software	128	57,4	57,9
Whether you use FS/OS or proprietary software is just a matter of what you want to do	72	32,3	32,6
Missing	2	0,9	

Tabelle 8: Einkommensverteilung

	Anzahl	%	Valide %
Less than US $ 250	31	13,9	14,6
US $250 to US $ 500	21	9,4	9,9
US $ 500 to US $ 1,000	25	11,2	11,7
US $ 1,000 to US $ 1,500	28	12,6	13,1
US $ 1,500 to US $ 2,000	19	8,5	8,9
US $ 2,000 to US $ 3,000	32	14,3	15
US $ 3,000 to US $ 5,000	29	13	13,6
US 5,000 to US $ 10,000	19	8,5	8,9
More than US $ 10,000	9	4	4,2
Missing	10	4,5	

Tabelle 9: Subjektive Einschätzung der eigenen Lebenssituation

	Anzahl	%	Valide %
I can live very luxriously	4	1,8	1,8
I can live very well	63	28,3	28,6
I live rather good	110	49,3	50
I just can pay my bills	36	16,1	16,4
I have problems to cover my basic costs	7	3,1	3,2
Missing	3	1,3	

Tabelle 10: Einstiegsmotivation

	Valid	Missing	Mean
I had a little problem, so I wrote nice little program	215	8	1,92
I had been a user of a program which I wanted to improve	220	3	1,94
To improve my programming skills	219	4	2,18
Programming seemed to be fun	218	5	1,74
To get in touch with others who have the same interests	211	12	2,9
It's a great feeling to see that people can create code voluntarily that is at least as good as that of big corporations	213	10	2,61
I wanted to be a hacker, because they seemed so cool	214	9	3,42
The form of cooperation	214	9	2,73
It's fun to get in touch with people from all over the world	210	13	2,94
To support in pratice the idea that information should be free	212	11	2,55
Here are people I can learn from	216	7	2,15
I wanted to give something back for all the FS/OS I used	219	4	2,16

Tabelle 11: Aktuelle Motivation

	Valid	Missing	Mean	+/-
I still have a little problem so I still write nice little program	213	10	1,85	+0,07
I'm a user of a program which I want to improve	218	5	1,54	+0,4
To improve my programming skills	212	11	2,21	+0,6
Programming is fun	217	6	1,65	+0,09
To get in touch with others who have the same interests	213	10	2,38	+0,52
It's a great feeling to see that people can create code voluntarily that is at least as good as that of big corporations	214	9	2,19	+0,42
Being a hacker is cool	213	10	3,19	+0,23
The form of cooperation	215	8	2,27	+0,46
It's fun to get in touch with people from all over the world	213	10	2,27	+0,67
To support in pratice the idea that information should be free	213	10	1,94	+0,61
Here are people I can learn from	216	7	1,89	+0,26
I want to give something back for all the FS/OS I use	219	4	1,82	+0,34