Entwickler Freier/Open Source Software
Abgrenzung durch Selbstdefinition

Frauke Lehmann

Freie/Open Source Software (FOSS) wird von Entwicklern aus aller Welt (meist) auf freiwilliger Basis geschrieben. Die sozialen Beziehungen zwischen den Entwicklern lokalisieren sich im wesentlichen im virtuellen Raum (Internet). Folglich haben die sozialen Interaktionen eine schmälere Bandbreite als im realen Raum.

Unter diesen Umständen ist die Stabilisierung des sozialen Zusammenhalts besonders wichtig, aber auch schwierig. Ein wichtiger Mechanismus ist der Aufbau einer kollektiven Identität, durch die sich die Entwickler auch nach außen abgrenzen. Die Abgrenzung der FOSS Entwickler geschieht auf zwei Ebenen, zum einen gibt es eine objektive, rechtliche Abgrenzung durch eine spezielle Form der Lizensierung der Software, zum anderen existiert eine subjektive, kulturelle. Diese Identität basiert auf dem Selbstbild der Entwickler. Zwar gibt es auch Feindbilder und Abgrenzung durch Distanzierung von anderen (meist Unternehmen im Computersektor), der Schwerpunkt der Identitätsbildung liegt aber auf der Selbstdefinition.

1 Lizenzen als objektive Grenze

FOSS unterscheidet sich von proprietärer Software durch eine andere Gestaltung der Eigentumsstruktur. Diese kann als grundlegende, objektive Definition des Feldes verstanden werden.

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 Bündelt lässt sich in vier Arten von Rechten aufteilen: Rechte des Gebrauchs, Rechte der Veränderung, Rechte der Übertragung und die Kompetenzkompetenz (Richter/Furubotn 1996: 82).[1]

Die Kompetenzkompetenzinhaber entscheiden über die weitere Verteilung der Rechte und können einen Teil ihrer Rechte an andere übertragen. Der Copyrightinhaber tritt entweder sein Copyright beispielsweise durch Verkauf an jemand anderes ab (inklusive der Kompetenzkompetenz) oder er gibt nur einzelne Verfügungsrechte durch Lizensierung weiter. An diesem Punkt unterscheidet sich FOSS von anderen Typen der Softwarelizensierung. Bei proprietärer Software sind die Rechte meist auf die Nutzung an einem einzelnen Rechner sowie eine einzige Sicherheitskopie beschränkt (vgl. z. B. Microsoft 1995).

Die FOSS Lizenzen haben dagegen eine deutlich andere Ausgestaltung bezüglich der Verfügungsrechte. Diese orientieren sich an den von Richard Stallman[2] definierten Freiheiten für FOSS (Stallman 1999: 56). Sie geben die Nutzung völlig frei, erlauben die beliebige Veränderung und geben die Verteilung der ursprünglichen wie der veränderten Versionen frei.

Einige FOSS Lizenzen haben ein Vererbungseffekt, der von Stallman als *Copyleft* bezeichnet wird. Der Copyrightinhaber nutzt dabei seine Kompetenzkompetenz, um denjenigen, die die Software verändern, vorzuschreiben, die gleiche Lizenz zu verwenden und so den Benutzern die genannten Freiheiten einzuräumen. Andere FOSS Lizenzen geben die Kompetenzkompetenz (fast) vollständig weiter.

[1] 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]).

[2] Richard Stallman ist der Begründer des GNU-Projektes (http://www.gnu.org) und der *Free Software Foundation* (FSF – http://www.fsf.org). Er hat FOSS als explizites Prinzip ins Leben gerufen und mit der *GNU General Public License* (GPL – www.gnu.org/copyleft/gpl.html) auch die bekannteste FOSS Lizenz entwickelt.

2 Subjektive Abgrenzung nach außen

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 FOSS Entwickler 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 (letzlich 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,[3] 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.[4] Zutreffend ist, dass Frauen einen verschwindenen Anteil ausmachen, 1,1 % sind nach den Ergebnissen der FLOSS-Studie (Ghosh et al. 2002:8) weiblich. Nicht zutreffend ist das Singledasein, nach der FLOSS Studie haben 58,5 %[5] eine Partnerschaft (ebd.: 11). Aus Texten über Linus Torvalds und Richard Stallman geht hervor, dass sie längere Zeiträume fast nur vor dem Computer verbrachten (Torvals/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 hinge-

[3] 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 Niktinzufuhr nicht leben bzw. arbeiten kann.

[4] 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).

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

wiesen, die dem dem Bild eines wahren *Geeks* oder *Nerds* entsprachen[6] 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 LinuxWorld Expo 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,[7] ist eine Teilnahme nicht möglich. Diese auf Wissen basierde 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".[8] 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 akuelle 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 ausgangen.

> „'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

[6] 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. 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.).

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

[8] „10" bedeutet im binärem Zählsystem „2". Gesehen auf T-Shirt.

> Hauptthesen ‚Nehmen Sie an, daß der Benutzer den IQ einer Kartoffel besitzt'. Dem werden wir sicherlich nicht folgen." (Lyx-Team 2002)

Äußerliche Attribute einer Person, wie Alter, Status (Titel) oder Aussehen scheinen unter den Entwicklern erst einmal irrelevant zu sein, so kommen beispielsweise auch Äußerungen wie „unser bester C-Programmierer war 13 Jahre alt" zustande.[9]

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.[10] Verstärkt wird die Abgrenzung durch eine eigene Sprache: Viele Begriffe stammen noch aus dem Jargon der Hacker vom *Artificial Intelligence Lab* (AI Lab) am *Massachusetts Institute of Technology* (MIT) (vgl. Levy 2001), aber es werden auch viele Unixbegriffe, insbesondere Kommandos, im normalen Sprachgebrauch verwendet, beispielsweise „grep" für durchsuchen.

Die oben beschriebenen Punkte haben FOSS Entwickler 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 von FOSS. Zunächst ist ein Einverständnis darüber vorhanden, dass Software frei sein sollte. Der Hintergrund dieser Überzeugung kann allerdings verschieden sein, so stehen für einige Entwickler ethische Gründe im Vordergrung, für andere sind dagegen arbeitspragmatische Erwägungen wichtig (siehe unten). Anders als bei Unternehmen, die Software durch den Verkauf bzw. Lizensierung als eine Möglichkeit Profit zu erzielen ansehen, steht bei FOSS, die Software als solche im Mittelpunkt.[11] Daher sind die Rahmenbedingungen

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

[10] „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 Scribts" sind kleine Programme zur Erledigung meist trivaler Aufgaben; „PEBKAC" bedeutet „Problem Exists Between Keyboard and Chair". Weitere T-Shirts sind unter anderem unter http://www.thinkgeek.com/tshirts/ zu finden.

[11] Profiterzielung vergleichbar proprieärer Software ist bei FOSS nicht wirklich realistisch, da eine Bedingung für die Freiheit der Software die Möglichkeit diese weiter zu verteilen ist. Zu anderweitigen wirtschaftlichen Nutzung FOSS siehe Raymond 2000b.

für FOSS 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 FOSS 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 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.[12] 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[13] umschreibt es ein wenig blumiger: „If software is elegant then the world smiles, the sun comes out, the clouds are fluffy and everything else.“[14] 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 Fußstafpen 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, inbesondere 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 Statusreduzieung, sondern es ist zunächst einmal eine Möglichkeit zum Lernen – für alle. So schreibt Gerog Greve, der Präsident der FSF Europe, dass „Für das technische Gelingen eines Projekts [...]

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

[13] 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.

[14] Interview am 24.6.2004 in Karlsruhe.

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.[15] Hinzu kommt, dass von den Beteiligten erwartet wird, bei einem Problem erst einmal selbst nachzudenken und sich im Zweifelfalle selbst kundig zu machen. So kommt es auch zu dem Ausspruch „RTFM – Read The F Manual“[16].

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.[17] 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

[15] Diese Einstellung liess sich gut an einer Auseinanderstetzung 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) 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.

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

[17] Eric Raymond ist selbst FOSS Entwickler. 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).

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.[18] Das Basarmodell zeichnet sich durch häufige Veröffentlichungen aus, damit die Entwickler von den Benutzern ständig wertvolle Fehlermeldungen bekommen können. Hierbei gilt der Satz, den Raymond das Linussche Gesetz getauft hat: „Given enough eyeballs, all bugs are shallow".[19] Auf diese Weise folgen die Veröffentlichungs-, Test- und Verbeserungsphasen 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).[20] Dieser Essay war für die Identitätsbildung der FOSS Entwickler 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,[21] so dass man sie mittlerweile als allgemeines Kulturgut und damit als Bestandteil ihres Selbstvestädnisses ansehen kann. Das Basarmodell wird heute (beinahe gleichwertig mit Stallmans Definition von FOSS) als Merkmal von FOSS verstanden.

[18] 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 Stellungsnahme 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).

[19] Nach Linus Torvals 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)

[20] 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).

[21] 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).

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 zugeschreiben werden und orientieren sich an deren Handlungsweisen. Die gemeinsamen Wurzeln lassen sich im Computerbereich finden, beispielsweise die Hacker vom MIT, die die Computer erforschen und deren Software optimieren wollten, oder auch die frühen Unix-User, die sich durch eine enge und freie Kooperation auszeichneten (vgl. Levy 2001; Salus 1995). Jon „Maddog" Hall geht sogar noch weiter in die Geschichte zurück, zu Alan Turing, Grace Murray Hooper, Maurice Wilkes.[22] Neben den Wurzeln gehören auch „öffentliche Personen" und Geschichten zu den gemeinsamen Traditionen. Beispiele sind hier die Auseinanderstetzung, die Richard Stallman Anfang der 1980er Jahre mit Mitarbeitern von Xerox hatte, da diese ihm nicht den Quellcode für einen fehlerhaften Druckertreiber geben wollten, damit er die Fehler beheben konnte (vgl. Williams 2002: 1ff.; Grassmuck 2002: 222), oder auch der Disput zwischen Linus Torvalds und Andrew Tanenbaum in der Minix-Newsgroup zu Beginn des Linux Projekts (o.V. 1999). Diese beiden Geschichten haben den Charakter und die Funktion von Gründungsmythen.

Neben diesen strukturellen Analysen und eher philosophischen Artikeln[23] gibt es mittlerweile einige Texte, die sich mit der endogenen Geschichtsschreibung beschäftigen, teilweise in Form von Autobiographien (beispielsweise Torvalds 2001a), die das kollektive Gedächtnis fördern.

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

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

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

Lizenzgebühren fordert. Allerdings veweigerte die SCO-Group lange einen Vergleich des AT&T-Codes mit dem Linux-Code und damit den Beweis, ob die Anschuldigungen überhaupt gerechtfertig 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". 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). 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 FOSS Geschäfte machen, es durch seine Unterstützung für die FOSS Entwicklung sogar geschafft, vom überlieferten Feind Nummer 1 (vgl. Young 1989: 381ff.; Levy 2001: 41f.) zum angesehenden Verbündeten zu bringen. Auch die als politisch eher radikal geltende FSF unterstreicht (Free Software Foundation 2004), dass der kommerzielle Handel mit FOSS zulässig und vollständig akzeptabel ist.

3 Zusammenfassung

Die FOSS Entwickler 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. FOSS unterscheidet sich von anderer Software durch die Gestaltung der verschiedenen Eigentumsrechte (Rechte der Nutzung, Rechte der Veränderung, Rechte der Weitergabe und die Kompetenzkompetenz).

Die zweite Form der Abgrenzung nach außen ist auf der kultutrellen Ebene angesiedelt. Neben dem mittlerweile positiv belegten Selbstbild als *Geek* oder *Nerd*, unterscheiden sich die FOSS Enwickler – 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.

Die Qualität und Detailiertheit dieser Selbstdefinition macht deutlich, dass sie die zentrale Rolle bei der sozialen Kohäsion unter den Entwicklern spielt. Erst aufgrund der geteilten Werte, Normen und Traditionen können Feindbilder überhaupt eine integrierende Funktion erfüllen. Sie stellen also nur eine verstärkende Versinnbildlichung dar.

Literatur

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 – Momopol. Frankfurt/Main: Suhrkamp, S. 103-129.

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.

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.

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.

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

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

Ghosh, Rishab Aiyer/Glott, RuedigerKrieger, 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.

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.

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 – Momopol. 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.

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. Aufl..

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

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.

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 Presantations.
http://www.linuxworldexpo.com/linuxworldny/V40/index.... am 22.1.2004.

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

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.

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.

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

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): 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.

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

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

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. Sebatopol: O'Reilly, S. 53-70.

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

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

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.

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.

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

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

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