Freie Software

Motivation und Engagement

Thomas Breitner

AUTHOR

Thomas Breitner

https://journalthing.thms.de

ISSN:

Freie Software

Motivation und Engagement

Magisterarbeit

zur

Erlangung der Würde des Magister Artium

der Philologischen, Philosophischen und Wirtschafts- und

Verhaltenswissenschaftlichen Fakultät der

Albert-Ludwigs-Universität

Freiburg i. Br.

vorgelegt von

Thomas Breitner

aus Sindelfingen

WS 2004/2005

Hauptfach Soziologie

Vorwort

Die vorliegende Arbeit beschäftigt sich mit einer Thematik, die dem Autor selber sehr am Herzen liegt. Dabei fällt der Autor nicht in die Kategorie des Entwicklers von Freier Software, er sieht sich vielmehr als Anwendern[1] und Unterstützer von Freier Software. Unter dieser Perspektive entstand diese Arbeit, die selbst auch komplett – inklusive der Transkription der Interviews – mit Hilfe Freier Software entstand[2].

Auch wenn die Thematik der Freien Software an sich nicht mehr jung ist, so findet doch erst in der jüngeren Vergangenheit eine intensivere Beschäftigung damit auch außerhalb des Kreises von Software-Entwicklung selber statt. Dieser Umstand impliziert zum Einen, daß das Thema für die Soziologie noch relativ unerschlossen ist, zum Anderen, daß der Großteil des Materials über dieses Phänomen sich dort findet, wo das Freie Software-Phänomen beobachtet werden kann: im Internet. Bedingt durch die – mögliche – Vergänglichkeit dieses Mediums ist sämtliche verwendete Literatur, welche aus dem Internet stammt, als pdf-Datei beim Autor dieser Arbeit auf Nachfrage zu beziehen[3].

Da diese Arbeit nicht auszuklammernde technische Hintergründe besitzt, ist entsprechendes Fachvokabular mit einem Symbol [→] gekennzeichnet und beim ersten Auftreten in einer Fußnote erläutert. Weiter findet sich unter Anhang I ein kurzes Glossar des verwendeten Fachvokabulars. Als Kurzreferenz für den geschichtlichen Hintergrund ist ebenfalls eine komprimierte Zeitleiste der Freien Software-Entwicklung unter Anhang II beigefügt.

Obwohl aus Gründen der Lesbarkeit im Text die männliche Form gewählt wurde, beziehen sich die Angaben auf Angehörige beider Geschlechter. Hiermit soll in keinster Weise eine geschlechterspezifische Differenzierung vorgenommen werden.

Einleitung

„Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs,
documenting his product and distribute for free?“

Rhetorische Frage von Bill Gates an die „hobbyists“; (Gates 1976, S. 1).

Relevanz

Alltag

Was sind das für „Hobbyists“, die ihr Wissen, ihre Fähigkeiten, ihre Zeit in die Entwicklung von Software investieren, um sie dann frei zu verteilen und anderen zur Weiterentwicklung anzubieten[4]? Was bringt diese Menschen dazu? Die Beantwortung dieser Frage bildet den Fokus der vorliegenden Arbeit.

Und in wie fern betrifft das Entwicklungsmodell dieser Gruppe von Hobbyists unseren Alltag oder unsere Erfahrung? Der Normalverbraucher frägt eventuell nach dem eigenen Nutzen, den

Berührungspunkten mit dem Phänomen Freier Software – und findet weder Nutzen noch Berührungspunkte. Freie Software, nur die Spielwiese von Technik-affinen Einzelkämpfern?

Nein. Freie Software ist einer der wichtigsten Spieler auf dem Software-Markt geworden, Freie Software-Entwicklung ist eine Herausforderung für gängige Organisationstheorien, Freie Software bedroht eines der größten Monopole dieser Welt und nicht zuletzt hat sie auch den Normalverbraucher erreicht – meist ohne, daß dieser dies mitbekommen hätte. Freie Software liefert dem Normalverbraucher die Antworten auf seine Fragen, die er beispielsweise bei http://google.de abgibt. Der Geist, der den befragten entfernten Computern Leben einhaucht, ist Freie Software[5], die Software, die dafür zuständig ist, daß man eine Webseite mit den Antworten zu sehen bekommt, ist Freie Software – über 67 % aller Webseiten werden von Freier Software ausgeliefert[6] – und nicht zuletzt ist fast alles, was dazwischen passiert, was mit den übermittelten Daten auf dem Weg über das Internet von einem Computer zum Anderen geschieht, mit Freier Software geregelt oder als Freie →Protokolle implementiert.

Validität

Nachdem sich Freie Software als ernstzunehmende Alternative zu proprietärer Software im wissenschaftlichen und ökonomischen Bereich entwickelt hat, beginnt sie seit einigen Jahren auch einen Siegeszug durch Politik und die Medien der Allgemeinheit anzutreten. Öffentliche und politische Institutionen wie beispielsweise der Bundestag, die Stadt München oder die Europäische Union machen sich für Freie Software stark. Die Gründe für dieses breite Engagement liegen in der Wirtschaftlichkeit der Verwendung von Freier Software, in der Sicherheit, die diese bietet sowie in der Unabhängigkeit von einzelnen Herstellern und auch in der Überzeugung, etwas Gutes zu tun. Auch die Printmedien haben das Thema entdeckt und damit eine breite, allgemeine Öffentlichkeit erreicht: Wurde 1997[7] noch lediglich sechs mal in nicht-spezialisierten Printmedien über Freie Software berichtet, so erschienen in den selben Medien[8] im Jahr 2003 über 400 Artikel. Auch in einem eher technischen Umfeld ist der ideologische Gegenentwurf, meist in Personalunion mit dem Microsoft-Konzern, der neuen Konkurrenz gewahr geworden[9] und kommuniziert diese Bedrohung auch nach außen in Form von Werbung[10] gegen Linux, dem wohl berühmtesten Vertreter Freier Software.

Eine neue Produktionsfunktion von freiwilligem Engagement

Das entscheidende Herausstellungsmerkmal von Freier Software ist nicht, daß sie kostenlos oder vergleichsweise billig zu haben ist, sondern die Produktionsfunktion, unter der Freie Software entsteht. Freie Software wird von dezentralen Netzen von Software-Entwicklern in ihrer Freizeit, wie auch im Dienste von Unternehmen geschrieben, gewartet und weiterentwickelt. Es ist das Verhältnis der Strukturmerkmale von proprietärer und Freier Software-Entwicklung, umrissen mit Schlagworten wie homogen versus heterogen, dezentral versus zentral, hierarchisch versus enthierarchisiert und proprietär versus frei, welches das Spannungsfeld um Freie Software-Entwicklung ausmacht. Ist die Produktionsfunktion des proprietären Modells geprägt vom Paradigma der individuellen Nutzenmaximierung, von Egoismus und Gewinnstreben als Quelle gesellschaftlichen Wohlstandes sowie der Orientierung an Angebot und Nachfrage, so basiert das zweite Modell – Freie Software – auf dem Gedanken der Kooperation und des freien Austausches, der im Grunde uneigennützigen Schaffung von öffentlichen Gütern. Der Begründer des Terminus „Freie Software“, Richard Stallman – und mit ihm viele Andere – gehen davon aus, daß Freie Software die Gesellschaft voranbringt, da sie die Anzahl der Güter, über die eine Gesellschaft frei verfügen kann, erhöht, und da sie ein neues Entwicklungsmodell als Exempel

auch für Bereiche außerhalb der Software-Entwicklung vorexerziert. Momentan ist dieses Entwicklungsmodell die einzig mögliche, dauerhafte Alternative zu den Produktionsfunktionen des freien Marktes – ihr Erfolg, die Qualität der Resultate und die enorme Anziehungskraft auf potentielle wie gegenwärtige Software-Entwickler scheint dem zuzustimmen.

Die soziale Dimension von Freier Software

Die soziale Dimension von Freier Software wurde bereits angesprochen: Sie wird verdeutlicht durch den Gegensatz zwischen der individuellen Nutzenfunktion herkömmlicher Software-Produktion, und der kollektiven Nutzenfunktion in Freier Software-Produktion. Entscheidend ist der soziale Kontext, in welchem Freie Software entwickelt wird, sowie die Interaktion der Freien Software-Community mit der sozialen Umwelt. Freie Software-Entwicklung ist eine soziale Bewegung – Organisation und Struktur sind mit anderen sozialen Bewegungen, wie der Umweltschutzbewegung, vergleichbar. Sie erfolgt in einem offenen, transparenten Prozeß aus der Kooperation der Gruppenmitglieder, welche einen Wertekanon – hier im Sinne von „frei“ und „offen“ – in Software teilen. Sie tritt aus diesem begrenzten Gruppenkontext hinaus und wird Teil der Umwelt der Gruppe als öffentliches Gut: Das Wissen eines Kulturraumes wurde neu zusammengesetzt, erweitert, und in den großen Pool an allgemein verfügbarem Wissen der Gemeinschaft wieder zurückgegeben.

Was bewegt die Freien Software-Entwickler, ihr Schaffen der Allgemeinheit zur Verfügung zu stellen und diese dazu zu ermuntern, sich dieses Wissen anzueignen und weiter zu transformieren? Wie in der Arbeit gezeigt wird, ist die Entwicklung von Freier Software ein Gruppenphänomen, und eine Reihe wichtiger Motivationen liegt primär in dem persönlichen Gewinn durch die Interaktion mit anderen Gruppenmitgliedern.

Aufbau und Zweck der Betrachtungen

Das Wissen über die Geschichte von Freier Software – und Softwareproduktion generell – ist ein essentieller Bestandteil für das Verständnis des Phänomens Freier Software und ist in Kapitel 2 ausgeführt. Darüber hinaus dient die historische Darstellung der Sensibilisierung gegenüber einem Software-Entwicklungsmodell, welches für den Großteil der Anwender im Laufe der Zeit durch Konzentration und Konsolidierung von alternativen, proprietären Systemen völlig aus dem Blickfeld verdrängt wurde und inzwischen in vielerlei Hinsicht nicht mehr als kontemporäre Realität wahrgenommen wird. Anhand der Ursprünge und Entwicklungslinien von Software-Produktion wird beispielhaft gezeigt, aus welchen Aspekten sich dieses Phänomen zusammensetzt.

Im Anschluß an die historischen Betrachtungen wird der Rahmen der vorliegenden Arbeit anhand gegenwärtiger Kontexte und der Relevanz von Freier Software aufgespannt (Kapitel 3). Dabei wird auf die Kategorie des Anwenders näher eingegangen, eine Kategorie, die durch die wechselseitige Beziehung zwischen Entwicklern und Anwendern im Freien Software-Modell Bedeutung erlangt. Die Freie Software-Gemeinde wird anhand empirischer Daten vorgestellt, um ein möglichst vollständiges Bild von den Menschen hinter Freier Software und ihrem sozialen Umfeld außerhalb der Softwareproduktion zu zeichnen.

Kapitel 4 stellt die Überleitung zur Untersuchung der einzelnen Motivationen dar. Es wird zum Einen auf die Methodik – insbesondere die qualitativen Interviews mit vier teilweise international renommierten Freien Software-Entwicklern – sowie die vorhandene Literatur eingegangen,

zum Anderen die Produktion von Freier Software vor dem Hintergrund wichtiger gesellschaftlicher Voraussetzungen und Tendenzen beleuchtet.

Im Hauptteil (Kapitel 5) folgt eine Analyse einzelner Motivationen für ein Engagement in Freier Software. Dabei werden vor allem die sozialen Dimensionen dieses Engagements gewürdigt sowie eine Kategorisierung aufgrund der sozialen Dimension der jeweiligen Motivation eingeführt. Es wird das Bedürfnis nach Anerkennung, Altruismus als Verhaltensdisposition, das Bedürfnis nach einem Gefühl der Zugehörigkeit sowie spezifische Wertevorstellungen als Motivationen detailliert beschrieben. Besondere Beachtung erfährt die spezifische Form des reziproken Altruismus, welche zur Erklärung bestimmter Verhaltensweisen in Freier Software besser geeignet ist als die Verwendung eines quasi idealtypischen Altruismus, wie er in der Literatur zu Motivation in Freier Software oftmals herangezogen wird. Da der Freiheitsbegriff eine wesentliche Rolle im Diskurs um Freie Software – in dieser Arbeit insbesondere im Zusammenhang mit der „ideologischen Motivation“ – einnimmt, ist ihm ein eigenes Unterkapitel gewidmet. Der Aspektes der Ideologie in Freier Software wird aus der Perspektive des Entwicklers, wie auch aus der Perspektive externer Unterstützer behandelt, da dieser Punkt die größte Schnittmenge des Freien Software-Entwicklungsmodells mit seiner Umwelt darstellt.

Es werden weiter Hedonismus, Mangel sowie das Bedürfnis nach Weiterbildung als ergänzende Motivationen behandelt.

Abschließend werden die zentralen Ergebnisse rekapituliert (Kapitel 6): Heterogenität der Motivationen, Relevanz ideologischer Motivation sowie die Bedeutung des Community-Aspektes unter der Perspektive einer spezifischen Form altruistischer Verhaltensdispositionen. Weiter wird ein Ausblick auf mögliche zukünftige Anknüpfungspunkte und Fragestellungen gegeben.

Historischer Überblick

Die historische Entwicklung von Freier Software ist ein Paradebeispiel für die enge Verzahnung von technischen wie sozialen Faktoren und der Progression eines in vielerlei Hinsicht von der Gesellschaft abgekapselten Teils von vergleichsweise kleinen, öffentlich kaum wahrgenommenen, aber dennoch bedeutsamen Zirkeln[11]. Die Arbeit konzentriert sich stark auf die Entwicklungslinien der UNIX-artigen →Betriebssysteme und dem näheren Umfeld wie etwa der Netzwerkfähigkeit. Der große Gegenpol im Sinne der Produktionsfunktion, der dahinterstehenden Ideologie[12] und der involvierten Personenkreise – das „proprietäre System Microsoft“ – wird, wenn möglich, ausgeblendet: einerseits aus Gründen des eingeschränkten Rahmens dieser Arbeit, andererseits, um die Grabenkämpfe und Glaubenskriege dieser beiden Lager und ihren Sympathisanten aus der Arbeit herauszuhalten.

Der geschichtliche Teil steht unter dem Paradigma des Umstiegs von analoger Datenverarbeitung auf digital implementierte Algorithmen, den Möglichkeiten der „Universalmaschine“[13], der Vernetzung von Rechnern und den daraus entstandenen Chancen für neue Kommunikationsmuster sowie neue Formen der Wissensverarbeitung und Wissensverbreitung. Ebenso ist die Diffusion dieser Paradigmen aus den elitären Technologiezentren[14] in den Alltag der Allgemeinheit, ihrer Interaktionsmuster, ihrem Erlebnisempfinden von Bedeutung.

Die Geschichte Freier Software wird im Folgenden anhand der wichtigsten Ereignisse und Eckdaten dargestellt[15].

Von UNIX zu Linux

Die Geschichte Freier Software beginnt mit der Entwicklung der ersten Computer und ist noch heute stark mit dem Pioniergeist der ersten Programmierer durchsetzt. In den Anfängen war Software kein eigenständiges Produkt wie wir es heute kennen, sondern eine Hinzugabe zu erstandener Hardware, eine Art „Betriebsanleitung“ für den erstandenen „Gegenstand“. Des weiteren waren diese Arten von früher Software auf fest definierte, abgegrenzte Einsatzbereiche zugeschnitten und noch dazu oft fehlerhaft.

Erwerben wir heute ein Software-Produkt „as-it-is“, also lediglich zur Anwendung, räumen uns durch den Erwerb quasi ein Nutzungsrecht ein, so gestaltete sich der Alltag in der Frühzeit der Computerisierung wesentlich komplizierter. Die Software mußte von dem Betreuer der entsprechenden Maschine erst angepaßt werden, damit diese überhaupt lauffähig war und den ihr zugedachten Zweck im Sinne des Erwerbers erfüllte. Dazu mußte eine Bedingung erfüllt sein, die noch heute einen Grundpfeiler von Freier Software darstellt: Das Recht darauf, zu studieren wie das Programm funktioniert und es gegebenenfalls an die eigenen Bedürfnisse anzupassen. Nur dadurch konnte gewährleistet werden, daß das Programm auf den noch nicht wie heute standardisierten Maschinen und der Vielfalt der unterschiedlichen →System-Plattformen lauffähig war. Auch wurde die Software von ihren „primären Herstellern“, den Architekten der entsprechenden Computer eher stiefmütterlich behandelt, da damals „kein Geld mit Software zu verdienen war“[16], was sich auch in der Qualität und dem Funktionsspektrum der Programme widerspiegelte. Um sich die aus heutiger Sicht undenkbar kleinen Dimensionen des damaligen Marktes für Soft- und Hardware – überspitzt – vorzustellen, sei ein sehr frühes Zitat aus dem Hause IBM angeführt, dem damaligen einzigen ernst zu nehmenden Anbieter für derartige Lösungen: „We anticipate a global world-market with place for perhaps five computers.“ (Thomas Watson, Chairman of IBM, 1943).

Das frühe UNIX

Die Geschichte des →Betriebssystems →UNIX ist in weiten Teilen wegweisend für die Entwicklung von Freier Software. Sehr viele der Tugenden und Traditionen, auf die sich noch heutige Freie Software-„Hacker“[17] berufen, entstanden im Dunstkreis des frühen UNIX. Dabei ist zu beachten, daß die Verwendung des Begriffs „→Hacker“ nichts mit der umgangssprachlichen Verwendung desselben gemeinsam hat, sondern Programmierer bezeichnet, welche sich mit Hingabe und Freude an der intellektuellen Stimulation dem Schreiben von Software widmen.

UNIX war die Fortführung eines zu groß angelegten und in den Augen verschiedener Zeitzeugen (Vgl.: Raymond 1999, S. 22) daher gescheiterten Betriebssystem-Projektes des →MIT sowie der Firmen General Electric und AT&T: „→Multics“[18]. Nachdem sich die Partner dieses Konsortiums trennten, führte der Informatiker Ken Thompson bei den Bell Laboratories die Arbeit an dem Projekt weiter, versuchte aber die teilweise falschen Designansätze zu revidieren. Die Bell Laboratories waren eine Forschungseinrichtung unter dem Mutterkonzern AT&T, dem damaligen Telekommunikationsmonopolisten der Vereinigten Staaten.

Thompson hatte – zusammen mit seinen Mitstreitern Dennis Ritchie und Douglas McIlroy – für die damalige Zeit hohe Anforderungen an das zu entwickelnde System gestellt: Es sollte Mehrbenutzerfähig (Multi-User) und damit Multitasking-fähig sein, was damals mit dem Begriff „time sharing“ umschrieben wurde und die bisherige Art der Aufgabenabarbeitung im seriellen Stil, wie beispielsweise durch das Lochkartensystem, ablöste. Weiter sollte das System aus

kleinen, einfachen „→Tools“ bestehen, deren geschickte Kombination die Ausführung komplexer Aufgaben ermöglichte. Somit bestand das System aus kleinen, gut wartbaren Komponenten – ein Umstand, der UNIX selber, aber auch dem →GNU-Projekt oder dem Linux-Kernel, zu einem sehr schnellen Entwicklungsmodell verhalf, da an verschiedenen Teilen des Systems relativ unabhängig gearbeitet werden konnte. Als sich dieses frühe UNIX – eigentlich: „→Unics“ – als tauglich für den produktiven Einsatz erwies, wurde es aufgrund von Überlegungen der →Portabilität in der Programmiersprache C, welche für die Arbeit an UNIX entwickelt wurde, komplett neu geschrieben. Bis zu diesem Zeitpunkt waren die meisten Programme in einer Maschinennahen Sprache wie →Assembler verfaßt, um das Maximum aus der begrenzten und teuren Ressource herauszuholen. Mit der steigenden Leistung und dem sinkenden Preis der Computersysteme rückten andere Aspekte in den Vordergrund – eines davon war die genannte →Portabilität. So war C eine „high level language“, deren Code erst durch das notwendige →Kompilieren in die auf die Zielhardware angepaßte Maschinensprache übersetzt und somit lauffähig wurde. Die zunehmende Vielfalt an Rechner-Architekturen zwang fast zu diesem „Zwischenschritt“, wollte man nicht jedes Mal – für jede Architektur – das Rad neu erfinden.

Bereits 1971 ist UNIX das Standardbetriebssystem im Mutterkonzern AT&T. Allerdings konnte AT&T keinen kommerziellen Nutzen daraus ziehen, da es dem Konzern aus kartellrechtlichen Gründen nicht gestattet war, sich außerhalb seiner Kerngeschäftsgebiete zu engagieren – und AT&T sah sich veranlaßt, den →Quellcode von UNIX zum Selbstkostenpreis an interessierte akademische Institutionen wie z.B. dem →MIT zu lizenzieren. Dieses Vertriebsmodell treffen wir noch heute bei den großen GNU/Linux-→Distributionen wie SUSE, Mandrake oder RedHat[19] an – nun gepaart mit wirtschaftlichen Interessen, welche aber nicht durch den Vertrieb, sondern durch Zusatzleistungen wie Support befriedigt werden.

Wie schon angemerkt wurde, ist in UNIX der Grundstein für den heutigen Erfolg und die Philosophie von GNU/Linux gelegt; technisch wie ideologisch[20] ist das frühe UNIX noch heute Vorbild für eine Reihe bedeutender Programmierer insbesondere von Freier, aber auch unfreier Software.

Diesen Teil der Geschichte von UNIX wird bei Grassmuck mit den passenden Worten „Die Kompaktheit, Modularität und Zugänglichkeit durch C [die →Programmiersprache C; Anm. d. Verf.] ermunterte viele Benutzer, eigene Entwicklungen durchzuführen, so daß UNIX schnell einen hohen Reifegrad erreichte“ (Grassmuck 2002, S. 212), abgeschlossen.

Der „zweite Teil“ der Geschichte von UNIX beginnt mit der Zerschlagung des AT&T-Konzerns 1984, wodurch es einem Teil des Konzerns ermöglicht wurde, die Quellen von UNIX öffentlich und kommerziell zu verwerten. Dieser Schritt führte zu einer ganzen Reihe von Abspaltungen vom „offiziellen“ UNIX-Codestamm, wie z.B. AIX von IBM oder HP/UX von Hewlett Packard und in seiner Folge – der Kommerzialisierung und der „Schließung“ des Codes – zu einer weitgehenden Unbrauchbarkeit für die akademische Beschäftigung an Universitäten an dem →Quellcode.

Würde man schon hier im Zusammenhang mit UNIX von „Freier Software“ sprechen, so wäre dies nicht völlig korrekt, da der definierte Terminus „Freie Software“ erst später (siehe Kapitel 2.1.4) offiziell Einzug in die Computer-Analen gefunden hat.

Berkeley UNIX

Während der Pionierzeit um UNIX bei den Bell Laboratories zeichnete sich bereits ein zweites Zentrum rund um die Forschung mit den UNIX-Quellen ab: Die Heimatuniversität von Ken Thompson, die University of California, Berkeley[21]. Durch die Verfügbarkeit des Quellcodes und die noch relative Überschaubarkeit dieses Codes war dieses junge, fortschrittliche UNIX prädestiniert dazu, von Forschern und Studenten in einem günstigen akademischen Umfeld unter die Lupe genommen zu werden. Hier war es möglich, an einem modernen und in seiner Architektur vollkommen neuen Produkt mitzuarbeiten – diese Möglichkeit wurde an der Universität von Berkeley intensiv genutzt. 1974 wurde das erste UNIX an der Universität installiert und von Studenten euphorisch angenommen.

Durch ihre Arbeit „in und an“ den Quellen von UNIX, der Behebung von Fehlern oder „Bugs“[22] und der eigenen Programmierung von als fehlend erachteter Funktionalität entstand bald – 1977 – ein eigener Zweig der UNIX-Entwicklung an der Universität Berkeley. Dessen Qualität und Feature-Reichtum inspirierte den damaligen Studenten Bill Joy dazu, die entstandene Software und Dokumentation in „handlicher Form“ zu einem Ganzen zusammenzustellen und an interessierte →Hacker zu verteilen. Fünf Jahre später wird Bill Joy das Unternehmen Sun Microsystems[23] gründen. Diese →Distribution von freier Software wurde „Berkeley Software Distribution“, kurz: →BSD, genannt.

TCP/IP

Es sollte sechs Jahre dauern, bis das →Betriebssystem UNIX im Jahre 1983 grundlegende Netzwerk-Funktionalität erhält. Die Netzwerkfähigkeit – ebenfalls an der Universität Berkeley entstanden – findet unter dem sperrigen Begriff →TCP/IP Einzug in den BSD-UNIX-Kernel und wird alsbald von allen möglichen weiteren UNIX-Versionen implementiert. Somit stellt das Jahr 1983 in zweifacher Hinsicht ein herausragendes Datum in der Geschichte von Freier Software im Speziellen und der Entwicklung von Computern im Allgemeinen dar: Zum Einen war die Möglichkeit der Vernetzung ein technologischer Meilenstein, der es erstmals ermöglichte, in Raum und Zeit unterschiedlich positionierte Rechner miteinander kommunizieren zu lassen, die Rechner als ein Kommunikationsmittel selber zu nutzen und damit zum Anderen auf sozialer Ebene die bisherigen Grenzen des „social hub“ an den verschiedenen Zentren der Computerentwicklung zu sprengen und in der Folge zu „entgrenzten social hubs“ zu mutieren. Der Economist schreibt in einem Artikel über Dennis Ritchie über das Bild des „hubs“ zur Zeit der Arbeit bei den Bell Labs bei AT&T treffend: „The technical hub of the system became the social hub.“ (Economist 2004, S. 2). In den Kapiteln 2.1.3 sowie 2.1.4 wird von einem Prototypen eines „social hub“ berichtet, eines – räumlich wie sozial – sehr eng gefaßten Umfeldes, in welchem Forschung, Arbeit und Begeisterung bezüglich des →Hackens stattfanden. Diese damaligen natürlichen Grenzen wurden mit der Möglichkeit der Vernetzung der Computer aufgehoben. Die tatsächlichen Möglichkeiten durch diese Vernetzung waren im Vergleich zu heute rudimentär, kein noch so phantasievoller, optimistischer und prophetischer →Hacker hätte damit gerechnet, daß heute beispielsweise →Voice-over-IP oder →WLAN zu bedeutenden Themen – auch außerhalb des technologisch-akademischen Publikums – werden könnten. Die ersten Netzwerkfunktionen bestanden aus einem simplen →UUCP und ermöglichten das Senden und Empfangen von einzelnen Dateien über reguläre Telefonleitungen – doch das allein war eine Sensation!

Die UNIX und die BSD-Chronik finden 1984 wieder einen gemeinsamen Bezugspunkt: Die Zerschlagung des AT&T-Konzerns in eine Reihe kleinerer, spezialisierter Unternehmen. Einem Software-orientierten Unternehmenszweig wurden die Verwertungs- und Lizenzrechte an den Quellen von UNIX zugestanden, was eine sofortige Änderung der Vertriebsart von UNIX zur Folge hatte: Da nun einer kommerziellen Verwertung nichts mehr im Wege stand, wurde UNIX zu einem regulär erwerbbaren Produkt, und die bisherige Praxis der Überlassung des →Quellcodes an bestimmte Institutionen wurde eingefroren. Auch die Universität zu Berkeley war von diesen Einschnitten in ihrem bisher sehr freizügigen Umgang mit den UNIX-Quellen betroffen. Um in gewohnter Weise – welche sich als sehr erfolgreich für die Universität erwiesen hatte – weiterarbeiten zu können, mußte das →BSD-UNIX so weit wie möglich von lizenzrechtlich bedenklichem Code befreit werden. Der Druck, diese Mammutaufgabe in Angriff zu nehmen, wurde noch durch einen Prozeß der AT&T-Tochtergesellschaft gegen die Universität zu Berkeley verstärkt. So wurde 1989 damit angefangen, diese bedenklichen, „unfreien“ Codeteile von Grund auf neu zu schreiben:

„[...] Bostic [Keith Bostic, seit 1986 aktives Core-Mitglied von Computer Systems Research Group; Anm. d. Verf.] pioneered the technique of doing a mass net-based development effort. He solicited folks to rewrite the Unix utilities from scratch based solely on their published descriptions. Their only compensation would be to have their name listed among the Berkeley contributors next to the name of the utility that they rewrote.“ (McKusick 1999, S. 42)

Dieser Prozeß wirkte sich nachhaltig auf die Entstehung von Freier Software aus und beschreibt prägnant das Entwicklungsmodell sowie eine der wichtigen Motivationen: Reputation (Vgl.: Kapitel 5.1). Durch diese hochgradig parallelen, netzbasierten Entwicklungsanstrengungen gelang die aufreibende Programmierarbeit, und letztlich konnte 1992 das erste voll funktionsfähige und freie →Betriebssystem mit einer Reihe von Zusatzsoftware als 386/BSD für den 386er PC von der Universität zu Berkeley veröffentlicht werden. Der große öffentliche Zuspruch, den das 386/BSD genoß, und die Flut an Rückmeldungen überstiegen bald die Kapazitäten der bisherigen Betreuer, und das Projekt wurde in Regie einer netzbasierten Gruppe, einer →Community, weiterentwickelt. Heute existieren drei unterschiedliche Entwicklungsreihen der →Berkeley Software Distribution, die eine aktive Community am Leben erhält, allerdings wurde die Popularität von →BSD durch das GNU/Linux-System (Kapitel 2.1.4 sowie 2.1.5) mitsamt seinem „freieren“ Entwicklungsmodell überholt.

RMS und die AI-Labs

1971 beginnt Richard Stallman[24]– vielen auch bekannt unter seinen Initialen und seinem Login-Name „RMS“ – seine Arbeit am →MIT Artificial Intelligence Laboratory als Systembetreuer und Programmierer. Damit betritt der Gründer des rechtlichen und ideologischen Konstrukts der Freien Software die Bühne der IT-Welt. Richard Stallman wird als euphorischer[25], brillanter[26] →Hacker charakterisiert, der sich mit großem Ehrgeiz und unter Entbehrungen[27] für das „freie“ Programmieren einsetzt. Steven Levy zeichnet in seinem mittlerweile klassischen Werk „Hackers. Heroes of the Computer Revolution“ aus dem Jahre 1984 ein sehr plastisches Bild von Stallman, welches zeitlich allerdings erst gegen Ende von Stallman's Engagement im AI Lab einzuordnen ist: „He considered himself the last true hacker left on earth.“ (Levy 1994, S. 427). Bis 1975 konnte Stallman seiner Lieblingsbeschäftigung ohne Einschränkungen nachgehen. Seine „Besessenheit“ wird oft an der Tatsache demonstriert, daß er sich seine Wahlberechtigungsunterlagen auf die Adresse seines kleinen Büros am →MIT ausstellen ließ, oft tagelang das Ge-

bäude nicht verlassen hat und auch in diesem Büro die Nächte verbrachte – meist vor dem Computer.

Der fruchtbare Nährboden des MIT seines frühen Schaffens prägte seine Einstellung zum Umgang mit Software und Computern nachhaltig. Er konnte forschen und entwickeln und hatte persönlich direkten Zugriff auf die Computer des →AI-Labs – ein Luxus in der damaligen Zeit. Er konnte innerhalb der Artificial Intelligence Laboratories mit den weiteren Mitarbeitern und Studenten eine regelrechte „→Hacker-Kultur“ aufbauen, die geprägt war von dem gegenseitigen und unbehinderten Austausch von Code und Wissen in einer unglaublichen Intensität. In der Literatur finden sich Beschreibungen zu dieser Zeit wie „Mekka der Computerwissenschaft“ oder „Hackerparadies“, jeweils mit der Betonung auf „Spaß“ als einem essentiellen Faktor zur Beschreibung der damaligen Arbeitskultur: „It was a bit like the Garden of Eden.“ (Stallman in: McHugh 1998, S. 2)

Dieses Hackerparadies begann 1975 auseinanderzubrechen. Inzwischen existierte ein kleiner, aber florierender privatwirtschaftlicher Software-Markt – im Gegensatz zu der an akademischen Institutionen wie dem MIT geschaffenen Software – der immer intensiver nach kompetenten Mitarbeitern suchte. Auf diese Weise verließen viele von Stallman's Kollegen und Freunden durch Abwerbung das Artificial Intelligence Laboratory, um gegen Entlohnung für die neu gegründete Firma „Symbolics“[28] zu arbeiten. Damit löste sich die für Stallman so fruchtbare Gemeinschaft oder auch „Community“ auf. Ein weiterer Schock für Stallman und seine Arbeitsweise war die Neuanschaffung eines Rechners, dessen Betriebssystem-Lizenz keine Modifikation des und keinen Austausch über den Quellcode erlaubte. An diesem Punkt stellte Richard Stallman schmerzhaft fest, daß sein Freiheitsbegriff nur noch eingeschränkt gültig war.

Noch vor diesem Schock begann Stallman mit der Arbeit an einem mächtigen Text-editor, der Genre Software, die das wichtigste Arbeitswerkzeug eines Software-Entwicklers darstellte. Stallman entschied sich aufgrund der guten Erfahrungen mit der Arbeitsweise an und mit den Computern des →AI-Lab, ein ähnliches Entwicklungsmodell für seinen Emacs, dem Texteditor, anzuwenden. Dies charakterisierte sich durch die freie Verfügbarkeit des Quellcodes und ist geprägt von Stallman's Wunsch, daß Entwickler, Anwender und Interessenten eigene Verbesserungen oder Erweiterungen vornehmen und diese an Stallman zurücksenden. Damit war das vermutlich erste bewußt initiierte, Community-basierte freie Software-Projekt entstanden. Grassmuck bezeichnet die Atmosphäre, unter welcher diese Software entstand als „sharing spirit“ (Grassmuck 2002, S. 219). Stallman trat als eine Art Moderator dieser offenen Entwicklergemeinschaft auf, der über die Aufnahme oder Ablehnung von Wünschen der weiteren Mitgliedern zu entscheiden hatte. Dieses Entwicklungsmodell erwies sich als sehr erfolgreich: Durch die enge Zusammenarbeit mit Anwendern und weiteren Entwicklern sowie der offenen, freien Verfügbarkeit des Quellcodes wurden nicht nur schnell vorhandene Fehler gefunden und behoben, sondern auch die Fähigkeiten des Programms stark und zügig ausgebaut.

Das GNU-Projekt

Nachdem das gewohnte und geliebte Arbeitsumfeld sowie die technischen Möglichkeiten um Stallman nicht mehr in gewohnter Form existierten, mußte sich Stallman zwischen zwei Alternativen entscheiden: Die erste wäre gewesen, sich dem Trend anzuschließen und ebenfalls unter der Regie eines Unternehmens proprietäre Software zu produzieren. Die zweite Alternative war, sehr offen formuliert, etwas anderes zu machen. Stallman's Liebe zum Programmieren hielt ihn davon ab, sich gänzlich in fremden Bereichen zu engagieren – er wollte weitermachen, und zwar wie bisher.

Daher setzte er sich ein großes, fast utopisches Ziel: Die Schaffung eines komplett freien →Betriebssystems mit allen nötigen Werkzeugen und Zusatzprogrammen. Stallman schreibt in der Ankündigung dieses Projekts, dem „GNU-Announcement“:

„Starting this Thanksgiving I am going to write a complete Unix-compatible software system called GNU [...], and give it away free to everyone who can use it. Contributions of time, money, programs and equipment are greatly needed.“ (Stallman 1983, S. 1)

Da ein derart umfangreiches Projekt nicht alleine zu schaffen war, sollte es in dem Schoße einer Community entstehen und reifen, wie es Entwicklungs-technisch der Emacs-Texteditor vorgemacht hatte. Die Architektur, die Stallman, der sich als starker und streitbarer Führer dieser Community erwies, auswählte, sollte sich sehr eng an dem →Betriebssystem UNIX anlehnen. Dieses wurde von Stallman als →portabel und vor allem als ein „sauberes Design“ angesehen. UNIX war damals das professionelle →Betriebssystem, im Gegensatz zu DOS oder frühen Windows-Versionen und wurde und wird hauptsächlich im Server- und Netzwerkbereich eingesetzt. Der UNIX-Philosophie folgend, sollte Stallman's Projekt sich nach und nach aus kleinen Komponenten – zum Beispiel →Tools wie dem →Texteditor, einem →Pager, einem →Kernel etc. – zusammensetzen, welche in ihrer Summe das angestrebte vollständige und freie →Betriebssystem ergeben würden.

Im Jahre 1983 gab Stallman seinem Vorhaben einen Namen, unter dessen Dach noch heute Freie Software-Projekte initiiert und gewartet werden: Das →GNU-Projekt. Der fremd anmutende Name entstammt der Sympathie von →Hackern zu Akronymen und bedeutet wörtlich: GNU's Not UNIX[29], womit die Unabhängigkeit vom ursprünglichen UNIX-Quellcode herausgestellt werden soll.

Um ein weiteres Zerbrechen dieser neuen Community zu verhindern, wie es Stallman bei der Abwerbung seiner Mitarbeiter erfahren mußte (Vgl.: Kapitel 2.1.3), und um die Arbeit der Community zu schützen, erfanden Richard Stallman und Eben Moglen[30] eine sogenannte „→Copyleft“-Lizenz. Diese Lizenz garantiert jene Freiheiten, welche Stallman nach dem Ende der erfolgreichen Arbeit an den Maschinen des MIT vermißt hatte. Diese Lizenz ist ein Meilenstein der Computer-Geschichte und wurde ebenfalls 1988 als „→GNU General Public License“ (GPL) in einem juristischen Rahmen verankert und veröffentlicht. Sie ist das Fundament, auf dem alle freien Softwareprojekte aufbauen und sie ist gewissermaßen die Vorlage für unzählige weitere – mehr oder weniger – freie Software-Lizenzen. Die darunter liegende Idee des →Copyleft steht ebenfalls Pate für eine ganze Reihe aktueller Entwicklungen in den Bereichen Intellectual Property und findet Einzug in die Verwertungsmöglichkeiten von künstlerischer Arbeit[31].

Stallman beschäftigte sich zu der Zeit mit den mehr oder weniger direkt angrenzenden Bereichen des „Hacking“ und verfaßt neben der GPL eine Reihe weiterer, inzwischen klassischen Schriften zu geistigem Eigentum, Freiheit in der Softwareproduktion und einer übergeordneten Ideologie der Freiheiten. Hervorzuheben ist vor allem das GNU-Manifest[32].

Im selben Jahr – 1985 – gründet Stallman die →Free Software Foundation[33], eine gemeinnützige Institution mit dem Zweck, Freie Software zu unterstützen und zu fördern sowie die Öffentlichkeit und betroffene Personenkreise gegenüber Freier Software zu sensibilisieren und über ihre Vorteile in Kenntnis zu setzen. Die Free Software Foundation ist noch heute eine der führenden und einflußreichsten Mitspieler auf dem Feld der Freien Software und besitzt inzwischen lokale Vertretungen in Asien und in Europa. Und immer noch, nach nunmehr fast 20 Jahren, ist

Richard Stallman, diese extrem kontrovers diskutierte Person, ihr unangefochtener Vorsitzender.

Damit ist der erste Abschnitt und der entscheidende historische Schnitt in der Art und Weise der Softwareentwicklung abgedeckt. Auch wenn schon hier einige mögliche Motivationen wie ideologischer Antrieb, technische Unzulänglichkeiten eines hinderlichen Entwicklungsmodells und natürlich die „Liebe“ an und der Spaß bei der Programmiertätigkeit durchscheinen, wird auf die Frage nach einzelnen Motivationen in den Kapiteln 5.1 bis 5.7 näher eingegangen.

Erst 1991 erreichen wir einen Punkt, in dem zum ersten Mal Begriffe und Namen auftauchen, die heute einer breiten Öffentlichkeit – auch jenseits der Technologie- und Hackertempel – bekannt oder sogar vertraut sind und welche oftmals als Synonym für Freie Software verwendet werden: Linux, das System und Entwicklungsmodell, und Linus Torvalds, „wohlwollender Vater“ und anerkannter „Führer“ eines der größten und bedeutendsten Freie Software-Projekte.

Ilkka Tuomi liefert uns die Überleitung zu Linux, indem er bemerkt, daß „[i]n this sense, Linux emerged exactly when it became possible.“ (Tuomi 2004, S. 18), und spielt damit auf die „Vorleistungen“ anderer Freien Software-Projekte – insbesondere dem →GNU-Projekt – sowie den technischen Voraussetzungen wie einem günstigen und auf breiter Basis verfügbaren Internet an.

Linux

Auch wenn das Projekt Linux oft in der klassischen Tradition von Freier Software beschrieben wird, so bedeutet dieses Projekt auch einen Bruch mit einer Reihe von den bisher behandelten Modellen und Initial-Faktoren. Erlebten wir Entstehung und Evolution von beispielsweise dem →GNU-Projekt oder dem →ARPANET als Produkte intellektueller oder technischer elitärer Zirkel, so kann bei Linux von einem „bottom-up“-Ansatz gesprochen werden. Aus dem Hobby eines „einfachen“ Studenten ging hervor, was von Vielen heute als Paradebeispiel für Freie Software angeführt wird – nicht nur was die Bedingungen der Entstehung, sondern auch was die Art des Umgangs und der Pflege des Codes wie auch der zugehörigen Community anbelangt. Die inzwischen oft gehandelte Gleichsetzung von Freier Software mit „Linux“ ist somit auch für einen Großteil der Mißverständnisse und falschen Einschätzungen bezüglich Freier Software im Allgemeinen verantwortlich.

Der Betriebssystemkernel Linux begann als Hobby eines finnischen Studenten in Helsinki. Dieser Student – Linus Torvalds – begann mit der Arbeit an Linux 1991, drei Jahre nach dem Beginn des Studiums der Computerwissenschaft. Was 1991 als kleines „Just for Fun“[34] -Software-Projekt begann, war noch in keinster Weise auf ein →Betriebssystem angelegt. Es war ein kleines Stück Software, welches Linus Torvalds' Leben etwas „erleichtern“ sollte. Seine Arbeit war inspiriert von dem damals zu Studienzwecken entworfenen Betriebssystem →Minix von Andrew Tanenbaum, welches an der Universität zur Veranschaulichung des UNIX-Betriebssystems gelehrt wurde. Minix besaß einige für Studenten sehr interessante Eigenschaften: Außer daß es das „große“ UNIX nachbildete, war es vor allem offen, das heißt es konnte der Quellcode studiert werden. Dies bedeutete, daß den Studenten die Möglichkeit gegeben war, zu erforschen wie dieses →Betriebssystem „im inneren“ funktioniert. Torvalds näherte sich sehr früh und sehr intensiv diesem Lehr-Betriebssystem, fügte neue Funktionalität hinzu oder spürte Fehler auf und behob diese. Immer mehr diente ihm Minix als Basis seiner Tätigkeiten am Computer. Eines seiner kleineren Softwareprojekte um Minix, ein →Terminalemulatorprogramm, gewann allerdings zunehmend an Bedeutung für ihn, sodaß im Laufe der Zeit auch Terminalemulator-fremde

Funktionen von Torvalds in das Programm implementiert wurden. Bald war das Programm in der Lage direkt auf Disketten und Festplatten zuzugreifen – was hier trivial klingt, war 1991 keineswegs eine Selbstverständlichkeit. Torvalds mußte einen Festplatten-Treiber und einen Disketten-Treiber von der Pike auf selbst schreiben. Diese „minimalistischen“ Anfänge sollten schon bald weit über ihr eigentlich intendiertes Ziel hinausschießen und zu dem werden, was wir heute als „Linux“ bezeichnen: Dem freien Betriebssystem-Kernel, der als letztes wichtiges Puzzlesteinchen des →GNU-Projektes gilt, der weltweit Rechnernetze, Serverdienste, Desktop-PC's oder →Smartphones zum Laufen bringt. Ganze Industrie- bzw. Dienstleistungszweige haben sich auf die Kombination der GNU-Software und Linux – GNU/Linux – „gestürzt“.

Linus' erste öffentliche Ankündigung des Projektes „Linux“ wird auf den 25. August 1991 datiert, den Zeitpunkt, an dem er in der →Newsgroup comp.os.minix das im folgenden zitierte Posting abgibt (Torvalds 1991):

Von: Linus Benedict Torvalds (torvalds@klaava.Helsinki.FI)

Betrifft: What would you like to see most in minix?

Newsgroups: comp.os.minix

Datum: 1991-08-25 23:12:08 PST

Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT protable [Fehler im Orig.; Anm. d. Verf.](uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(.

Von Anfang an war Linux ein gemeinsames Produkt seiner wechselnden Mitarbeiter und →Kontributoren und wird bis heute von dem „wohlwollenden Diktator“[35] Linus Torvalds geleitet.

Sehr bald fanden sich genügend Studenten und anderweitig Interessierte, die durch die nun verschlossenen UNIX-Quellen sowie der Unzufriedenheit mit der Art der Weiterentwicklung von Minix, an Linux, diesem „wirklich“ freien Projekt, mitarbeiteten. Getragen von den technischen Möglichkeiten der Kommunikation über hauptsächlich universitäre Netze und der Begeisterung über die Arbeit an einem derart „entfesselten“ Software-Projekt, machte Linux, der erste tatsächlich nutzbare →Kernel für das GNU-System, große Fortschritte. Ein „Wettlauf“ der Versionsnummerierung beginnt und hält bis heute an. Die Dimension dieser Entwicklung läßt sich anhand der Größe des Quellcodes oder der Versionsnummern ablesen: Begann Linux mit der Version 0.01 im Jahre 1991 mit 8.413 Zeilen Code, so ist der aktuelle Stand – im Juli 2004 – ein als stabil veröffentlichter →Kernel 2.6.7 mit 5.499.597 Zeilen Code[36]. Wird heute oft der Code von

Linux als „genialer Hack“ bewundert und gelobt, so ist es, um es mit Eric Raymond's Worten zu sagen, ein ganz anderer Aspekt, dem diese Würdigung in erster Linie zukommt:

„In fact, I think Linus' cleverest and most consequential hack was not the construction of the Linux kernel itself, but rather his invention of the Linux development model.“ (Raymond 2001a, S. 27)

Der Geschichte des Linux-Kernels wird in der Literatur wesentlich mehr Platz und Würdigung eingeräumt, zum Einen weil →Linux heute eines der meist-beachteten Projekte Freier Software ist, zum Anderen auch, da es unzählige Anekdoten, Zufälle und Kuriositäten im Zusammenhang mit der Entstehung und Entwicklung von Linux zu berichten gibt. Dies kann und will ich an dieser Stelle nicht in allen Details wiedergeben. Auf einzelne Aspekte werde ich in den folgenden Kapiteln ausführlicher eingehen, etwa die Community-basierte Zusammenarbeit (siehe Kapitel 5.3) oder das Gefühl eines „Mangels“ (siehe Kapitel 5.6) – technologischer wie ideologischer Art – welcher motivierend in Richtung seiner Beseitigung wirkt. Inzwischen – Linux ist nun schon über 10 Jahre „alt“ – ist seine Geschichte auch von zu vielen roten Fäden durchzogen, als daß man diese Chronik auf ausgewählte Entwicklungslinien herunterbrechen könnte. Unter dem Schirm von Linux findet sich das gesamte Spektrum möglicher Motivationen: Von Protesthaltung bis Spaß, von monetären Anreizen bis zu einer viel allgemeineren Ideologie der Freiheit[37].

Zwischenfazit zur Geschichte Freier Software

Wie wir gesehen haben, läßt sich die Chronik Freier Software nicht auf einer linearen Zeitleiste abbilden. Vielmehr greifen zu ihrer Charakterisierung Begriffe, welche auch zur Beschreibung von Freier Software selber herangezogen werden: So kann man sich diese Geschichte Freier Software ebenfalls als etwas Verteiltes, nicht direkt Verbundenes oder gar Koordiniertes vorstellen. Sehr plastisch bringt es eine Analogie zum in Kapitel 2.1.4 beschriebenen →GNU-Projekt zum Ausdruck: Diese Geschichte als großes Puzzle, an dessen Ecken und Enden jeweils unterschiedliche Akteure mit der Zusammensetzung anfangen, die sich, solange es noch nicht zum Zusammenfügen bereits fertiger „Puzzle-Inseln“ kommt, weitgehend autonom verhalten können und eine Koordination erst mit dem Zusammensetzen des „Ganzen“ nötig wird. Die „Architektur“ des Puzzles war weitgehend durch das Design des UNIX-Systems und dem →POSIX-Standard vorgegeben, und wichtige Komponenten dieses Puzzles von Projekten wie →XFree86 und den GNU-Programmen[38] abgedeckt. Aus der technologischen wie der historischen Perspektive war es Linux, der →Kernel, das letzte fehlende Puzzle-Teil, welches in ein sonst schon fertig zusammengesetztes Puzzle eingefügt werden mußte und Freie Software zu einem Ganzen verwandelte. Letztlich konnte dieses fertige Puzzle als stabile Basis für weitere Entwicklungen und Visionen dienen, welche weit über den Horizont von Software und →Betriebssystemen hinausragen[39]. Richard Stallman formulierte diesen Umstand auf der WOS I-Konferenz[40] folgendermaßen: "The core of the GNU project is the idea of free software as a social, ethical, political issue: what kind of society do we want to live in?" (Stallman 1999a, S. 2).

Gegenwärtige Kontexte und Relevanz

Haben wir bisher in der Geschichte von Freier Software nicht zwischen Anwendern und Entwicklern unterschieden, so bedarf diese „Vermengung“ einer genaueren Betrachtung, die für die weiteren Auseinandersetzungen mit dem Thema Motivation und Engagement in Freier Software

essentiell ist. Der in Kapitel 2 dargestellte historische Abriß von Freier Software und Softwareentwicklung endet mit den Ausführungen zu dem Beginn von Linux, dem →Kernel des GNU/Linux-Systems. Dies liegt inzwischen über 10 Jahre zurück, was im IT-Sektor einer „halben Ewigkeit“ gleichkommt. Vergleiche zwischen einer damals und heute aktuellen Betriebsumgebung in Soft- wie Hardware verdeutlichen diese Diskrepanz der Dimensionen.

Ähnlich verhält es sich mit den „Produktionsbedingungen“ von Software. Die Umschreibungen der damaligen Situation können heute nicht oder nur noch sehr eingeschränkt benutzt werden (Vgl.: 2.1.3). Mit einer exponentiell gestiegenen Bedeutung und einer im Vergleich zur Zeit vor 10 Jahren sehr viel breiteren Anwendung von Computern und Software ist auch dieser Bereich einem tiefen Wandel unterzogen. Die für diese Arbeit vielleicht wichtigste Wandlung ist die Spaltung dieses Mikrokosmos in Anwender und Entwickler – zwei Kategorien, welche in den Anfängen der Softwareproduktion meist noch in Personalunion repräsentiert waren. Diese Trennung ist nicht künstlich, sie ist auch nicht konkret datierbar; sie ist vielmehr eine Folge eines immer weiteren Eindringens von in Software realisierten Systemen in immer alltäglichere Bereiche: Software als Massenprodukt. Immer mehr Menschen „nutzten“ als Anwender bestimmte Produkte, welche von anderen – den Entwicklern – entwickelt wurden und eine der wichtigen Feedback-Schleifen[41] war – zumindest auf breiter Ebene – durchbrochen. Die folgenden Ausführungen (ab Kapitel 3.1) beziehen sich auf den Entwickler.

Eine Ausdehnung auf die Anwender - im Folgenden verstanden als reine Nutznießer einer bestimmten Technik, welche sich nicht an der Entwicklung oder der Kommunikation mit den Entwicklern des jeweiligen Produktes beteiligen – würde den Rahmen dieser Arbeit sprengen. Auch inhaltlich stehen bei Anwendern bezüglich der Motivation für Freie Software andere Aspekte im Vordergrund, die sich sehr oft auf wirtschaftliche und politische Überlegungen herunterbrechen lassen[42] und hier nicht behandelt werden. Allerdings nimmt der Anwender im Produktionsprozeß von Freier Software eine besondere Stellung ein, daher sei im Folgenden kurz darauf eingegangen.

Da es sich um ein symbiotisches Abhängigkeitsverhältnis der beiden Kategorien Anwender und Entwickler handelt, sei hier kurz die Dimension „Anwender“ angerissen. Als Anwender versteht man im klassischen Sinne natürliche wie auch juristische Personen, worunter Privatnutzer als auch Organisationen oder Firmen fallen, welche eine bestimmte Nutzung zwar für ihre Bediensteten veranlassen aber selbst nicht als „Benutzer“ in Erscheinung treten. In der entsprechenden – auch deutschsprachigen – Literatur wird oft synonym der Begriff „User“ verwendet. Wie schon angedeutet, ist „Anwender“ als das Gegenstück von „Entwickler“ zu verstehen. Umgangssprachlich steht „Anwender“ allerdings oftmals für den privaten, heimischen Nutzer eines Rechners[43].

Allein die unterschiedlichen Interessen dieser heterogenen Gruppe von Anwendern macht eine befriedigende Analyse bezüglich der jeweiligen Motivationen für die Nutzung von Freier Software in diesem Rahmen unmöglich[44].

Dimension von „Freiheit“ und Software

In der vorliegenden Arbeit sind nicht die Motivationen der Anwender von Interesse, die – um nur einige zu nennen – von finanzieller Entlastung bis zu post-moderner Pseudo-Rebellion[45], von der Suche nach Unabhängigkeit bis zu Sicherheitsaspekten reichen[46].

Vielmehr sind es letztlich die Anwender, die dem Phänomen Freie Software Bedeutung verleihen. Ebenso wie es für ein Freies Software-Projekt (Vgl.: Kapitel 5.3) von essentieller Bedeutung ist, eine Community um das Projekt aufzubauen, so ist es für die Zukunft von Freier Software wichtig, daß sie genutzt wird. Nur so tritt sie aus der Sphäre der →Hacker heraus und durchdringt weitere Bereiche[47]. Man kann somit anhand der quantifizierbaren Menge an Anwendern, Befürwortern und Unterstützern von Freier Software eine „Bedeutungs-Dimension“ aufspannen, eine Dimension, die verdeutlicht, wie wichtig das Thema Software und Freie Software im Speziellen in einer von Informationsverarbeitung bestimmten Welt ist – und daß Freie Software ein fester Bestandteil kontemporärer Realität[48] geworden ist.

Dies ist der Punkt, mit dem den Fragen nach der sozialen Dimension in Freier Software begegnet wird: Versteht man Soziologie als die Erforschung von sozialen Regeln und Prozessen, welche die Menschen verbinden oder trennen - und zwar nicht auf der Ebene des Individuums, sondern die speziell auf das Individuum als Mitglied einer sozialen Gruppe oder einer Institution abzielt - so ist Soziologie heute auch immer mehr eine gesellschaftliche Komponente, welche über Software katalysiert wird oder werden kann. Klassische Domänen von Soziologie sind direkt von den Auswirkungen einer zunehmenden und tieferen Durchdringung des gesellschaftlichen und zwischenmenschlichen Alltags durch Software betroffen. Dies geht weit über die auf den ersten Blick offensichtlichen neuen Kommunikationschancen und Kommunikationsbarrieren hinaus. Eine Realität, welche von abstrakten, durch Netze vermittelte Prozesse am Leben erhalten wird, entzieht sich mehr und mehr seiner Basis. Diese auseinanderdriftende Kluft zwischen der „erlebten Realität“ und dem abstrakten, nicht-trivialen Regelwerk, dem diese Realität gehorcht, stellt eine Herausforderung an jede moderne Gesellschaftsform dar. Die oft beschworenen und in Ansätzen schon umgesetzten Vorteile aus einer solchen Realität wie erweiterte Unabhängigkeit von Zeit und Ort – beispielsweise konkret umgesetzt in Bürgernähe und Kommunikationschancen – bringen eine ganze Reihe möglicher Risiken mit sich, deren öffentliche Diskussion gerade im Zusammenhang mit Freier Software wieder auflebt.

Um die Relevanz der Thematik von Freier Software konkret aufzuzeigen, seien zwei Beispiel angeführt:

Offenes Kommunikationsmedium

Email wird als ein selbstverständliches Trägermedium für Nachrichten an einen idealtypisch nicht begrenzten Personenkreis benutzt. Man geht davon aus, daß jeder, dem man eine Email schickt, diese auch „öffnen“, kann. Wir stellen uns die darunterliegende Infrastruktur nicht als ein geschlossenes System mit bestimmten Grenzen vor – daß beispielsweise die Email an nationalen Grenzen Halt macht oder deren Auslieferung an vorgezeichnete Vertriebswege wie innerhalb eines lokalen, sagen wir universitären Postnetzes, gebunden ist. Ferner ist es für uns irrelevant, „mit was“ – also mit welcher Software, auf welchem →Betriebssystem und welcher Art von Hardware – der Adressat unsere Nachricht „öffnet“. Aber: Alle diese unterstellten Selbstverständlichkeiten sind nicht selbstverständlich. Sie sind eine Konsequenz der offenen Architektur des Internet, das nicht im klassischen Sinne monopolistisch beherrschbar ist, sowie einer ebenso offenen Konzeption der für das Medium Email notwendigen →Protokolle und →Schnittstellen. Nur diese Offenheit ermöglicht die Freiheit, die wir im Umgang mit dem Medium empfinden. Ein Gegenentwurf hierzu wäre ein proprietäres Email-Protokoll[49], welches nur mit einer bestimmten Software, eventuell nur in bestimmten Ländern und von bestimmten Personenkreisen genutzt werden kann; oder deren Nutzung an zusätzliche Lizenzkosten oder

Nutzungsgebühren gebunden ist – wie wir es aus der klassischen Telefon-Kommunikation oder dem Postwesen kennen.

Dieser Gegenentwurf ist keine „unmögliche Utopie“, sondern eine realistische Alternative wenn man die bedeutenden Akteure auf der Ebene des Personal Computing und des Internet betrachtet. Diese „Utopie“ soll von zwei Paradigmen bestimmt sein: Zum Einen von dem Gleichnis der Sprache – auf der Ebene der zwischenmenschlichen Kommunikation sowie dem →Protokoll, der technisch realisierten Kommunikation auf Rechner-Ebene – zum Anderen von der Kontrolle (s.u.).

Das Gleichnis der Sprache soll den Aspekt der freien Nutzung verdeutlichen: Wir nehmen es als selbstverständlich an, daß wir uns mit Dritten sprachlich verständigen können, solange wir der selben Sprache mächtig sind. Wir betrachten weiter Sprache nicht nur als ein Distinktionsmerkmal gegenüber dem Tierreich, sondern vielmehr als die Voraussetzung für unsere kulturelle Evolution. Weiter gehen wir von einer „Freiheit der Sprachen“ aus. Wir dürfen uns frei aller denkbaren Sprachen bedienen, diese lernen und nutzen und sehen uns in deren Verwendung nicht künstlich beschränkt oder kontrolliert. Übertragen wir dieses Bild auf den Bereich Computer und Internet, so finden wir eine ganze Reihe von Parallelen, aber auch einige „Unstimmigkeiten“. Dabei müssen wir unterscheiden, was diese Kommunikationsform – die „Sprache der Computer-Netze“ – ist und was sie sein kann. Hier kommt der zweite oben genannte Punkt ins Spiel, die Kontrolle. Eine „öffentliche Sprache“ darf nicht durch bestimmte Institutionen oder Unternehmen kontrolliert werden. Übertragen auf die Sphäre der Computer müssen demnach die →Protokolle, über welche Computernetze kommunizieren für jeden einsehbar und ohne Restriktionen in eigene Projekte implementierbar sein. Dies ist in etwa die momentane Situation für die wichtigsten Protokolle wie →http, →ftp, →smtp, →pop und →imap. Der „utopische Gegenentwurf“ allerdings könnte folgendermaßen aussehen[50]: Eine Firma mit einer Monopolmachtstellung auf verschiedenen Gebieten des Computing führt ein eigenes, proprietäres Protokoll – beispielsweise für die Email-Kommunikation ein – und dank der Marktdurchdringung von bis zu 95 % auf Desktop-Betriebssystemen entwickelt sich dieses Protokoll für die Nutzer dieser →Plattform schnell zum Quasi-Standard der Kommunikation mit anderen Teilnehmern der Plattform. Die Konsequenzen könnten so aussehen, daß Nutzer anderer Plattformen für den Anschluß und die Nutzung dieses Protokolls – bzw. dieser „Technologie“ – Entgelte in Form von Lizenz- oder Nutzungsgebühren an den Technologie-Inhaber abführen müßten. Diese Kosten könnten an den Endbenutzer, also den tatsächlichen Kommunikator[51] weitergereicht werden. Dieser Fall besitzt zwei wichtige Implikationen: Zum Einen muß die Lizenz nicht gewährt werden; es wären somit bestimmte potentielle Nutzer ausgeschlossen, zum Anderen drängt diese Strategie weitere Wettbewerber – oder offene Protokolle – soweit ins Abseits, daß sich deren Betrieb nicht mehr lohnt und das Quasi-Monopol sich zu einem tatsächlichen Monopol verwandelt. Eine zweite Frage entsteht abseits der wirtschaftlichen Auswirkungen und möglichen Monopol-Schäden und betrifft die Herrschaft[52] über eine Kommunikationsform: Der „Inhaber“ eines proprietären Kommunikationsprotokolls ist der Einzige, der die Funktionsweise dieses Protokolls vollständig „versteht“, er ist ebenfalls der Einzige, der dieses Protokoll – ohne Wissen der Nutzer – manipulieren und verändern kann. Dies hat weitreichende Konsequenzen. So sind Situationen denkbar, daß die über dieses Protokoll laufende Kommunikation auf bestimmte Muster hin untersucht wird und diese so – ohne Wissen des Nutzers – gewonnenen Daten weiterverarbeitet werden; ob dies im Sinne staatlicher Überwachung oder kommerzieller Ausspähung von Konsumverhalten geschieht ist zweitrangig – entscheidend ist die Möglichkeit dazu[53].

Die heutige Situation sieht anders aus: Die wichtigsten Protokolle sind „frei“, und unabhängige Institutionen pflegen diese in einem transparenten Prozeß[54]. Die sehr frühe Durchdringung des Internet mit diesen Standards, welche in sogenannten „→Request for Comments“[55] niedergelegt sind, sorgte dafür, daß das Monopol auf der Seite der offenen Standards war und bis heute ist. Die Vorteile können wir heute genießen, indem wir – ohne zu wissen, wo unser Kommunikationspartner sich aufhält oder welche →Email-Client-Software[56] er nutzt – mit ihm via Email in Kontakt treten können – und dies in den meisten Fällen problemlos funktioniert.

Elektronisch konstituierte Machtverhältnisse

Dieses Beispiel gesellschaftlicher Relevanz betrifft die Verbindung von „freien“ Strukturen und demokratischen Legitimationsprozessen. Öffentlichkeitswirksam dargestellt und kontrovers diskutiert anhand des US-amerikanischen Dilemmas mit elektronischen Wahlmaschinen. Hervorzuheben ist die Aktualität der Thematik, die erstmals auch in den konventionellen Printmedien für eine breite Öffentlichkeit dargestellt wird und somit mehr Menschen für das Thema „offene, freie Standards“[57] sensibilisiert. An dieser Stelle folgt lediglich eine knappe Zusammenfassung, für eine ausführliche Darstellung sei auf Kapitel 5.4 verwiesen, in welchem eine bestimmte Ideologie als Motivation für Freie Software behandelt wird und zusätzliches Material (siehe Fußnote 182) angegeben ist.

Als George Walker Bush 2001 zum 43. Präsidenten der Vereinigten Staaten von Amerika gewählt wurde, kam es zu Unregelmäßigkeiten bei dem technischen Verfahren dieser Wahl. Erstmals wurden bei Präsidentschaftswahlen elektronische Wahlmaschinen zur Stimmabgabe eingesetzt. Während und nach der Wahl wurden Vermutungen laut, die behaupteten, daß ihre Stimme aufgrund technischer Probleme der Wahlmaschinen nicht in die offizielle Zählung eingegangen seien – interessanterweise immer zu Gunsten von George W. Bush's Kandidatur. Die Maschinen wurden von einem privat-wirtschaftlichem Unternehmen betrieben, deren Software eigentlich einem behördlichen, unabhängigen Review-Prozeß unterworfen sein sollte. Diese behördliche Kontrolle versagte jedoch nachweislich in einigen Punkten. Die Details sind hier nicht von Bedeutung, wohl aber die abgeleiteten Konsequenzen und Forderungen, da diese sich in Idee, Motivation und Formulierung mit entsprechenden Aspekten auf der Seite Freier Software deckt.

Die „Idee hinter der Kritik“ ist – formuliert mit den Worten von Marco Schulze – im Folgenden dargestellt. Das folgende Zitat erlangt dadurch zusätzliche Brisanz, da es vier Monate vor der aktuellen Präsidentschaftswahl im Jahre 2004 aufgezeichnet wurde.

„Wenn man sieht, wie wichtig Software für unsere Gesellschaft ist, dann muß man einfach sehen, daß es für unsere Gesellschaft notwendig ist, daß bestimmte Software auf jeden Fall frei ist. Zum Beispiel denke ich da an E-Voting-Systeme. In fünf oder zehn Jahren wird kein junger, informatisch gebildeter Mensch zu einer Wahlurne persönlich laufen, es sei denn er ist der Ansicht er brauche etwas Bewegung. Die Wahl über das Internet wird in ein paar Jahren gang und gäbe sein. Wenn man sich jetzt vorstellt, das E-Voting-System kommt dann von einer Firma, egal welcher, kann sie in dieses System reinschreiben was sie will. Zum Beispiel: Wenn ich dieses System schreiben würde, und ich wüßte ganz genau daß ich die Quelltexte nicht veröffentlichen müßte, dann würde ich sicherlich dafür sorgen, daß die CDU nicht so viel Stimmen wie die Grünen bekommt. Gut, das war jetzt nicht so ernst gemeint, aber ich meine: Wer kann das überprüfen, solange die Quelltexte nicht offen sind. Das zieht sich ja durch das ganze System durch: Ich benutze den Internet Explorer auf Windows, wer sagt denn, daß der Internet Explorer an das Formular wirklich das überträgt was ich eintrage? Also letztlich ist dafür not-

wendig, das meiner Meinung nach möglichst viel, möglichst die gesamte Kette dieses E-Voting-Systems vollkommen freie, quelloffene Software ist in die jeder Einblick hat. Denn dieses E-Voting-System hat letztlich ein Paradoxon als Auftrag: Es muß einerseits anonymisieren, denn es darf bei der Auswertung nicht mehr ersichtlich sein, daß Hans-Otto Partei XY gewählt hat, andererseits muß es aber sicherstellen, daß jeder nur einmal wählt - es muß aber auch gleichzeitig sicherstellen, daß die Stimmen noch einmal ausgewertet werden können wenn irgendwelche Ungereimtheiten auftreten. Das ist eine sehr komplizierte Aufgabe. Und allein durch die Tatsache daß es anonymisieren muß heißt das, daß die Datenbestände, die dieses „Ding“ verwaltet auf jeden fall unsichtbar, also versteckt sein müssen. Wenn ich aber keinen direkten Einblick in die Datenbestände haben darf, allein schon als Grundvoraussetzung, dann ist es natürlich zwangsweise so, daß ich dann Zugriff auf die Quelltexte haben muß um zu sehen, was dieses „Ding“ mit diesen Datenbeständen macht. Das ist etwas, das nur durch Freie Software gewährleistet wird.“ (Schulze 2004, S. 13f)

Exkurs: Der Hacker

Da in der Vorbereitungsphase dieser Arbeit sehr oft danach gefragt wurde, was für Menschen sich denn hinter dem Begriff „Freie Software-Entwickler“ verbergen, sei hier kurz das vorhandene, teilweise empirisch fundierte Material vorgestellt sowie eigene Gedanken dazu vorgetragen.

Spektrum

Richard Stallman ist aus der Perspektive des „Normalverbrauchers“ ein klassischer Repräsentant der →Hacker-Kultur. Das Stereotyp des technischen Experten und gleichzeitig sozial Zurückgezogenen scheint auf den ersten Blick bei Dr. Stallman zuzutreffen. Diese sehr enge Sichtweise trifft dennoch auf Richard Stallman nicht zu. Zur Falsifikation einer solchen Aussage ziehe man seinen „Crusade [dt.: Kreuzzug] for Free Software“ (Williams 2002, S. 1) heran, ein technologisches wie ideologisches Überzeugungs-Mammutprojekt, welches Dr. Stallman noch heute – beispielsweise während des Sommers 2004 in den Ländern des Baltikums – führt und für seine Botschaft den direkten Kontakt zu seinem Publikum sucht. Ein Publikum, das schon lange nicht mehr ausschließlich aus Experten besteht. Trotzdem markiert Dr. Stallman einen der Pole des Spektrums der Programmierer – oder →Hacker. Er hat sein Leben ganz dem „Crusade for Free Software“ (ebd.) gewidmet und verzichtet zugunsten dieses Zieles auf eine ganze Reihe „profaner Vergütungen“ wie Familie oder finanziellen Reichtum.

Der andere Pol ist mit Linus Torvalds besetzt, welcher eine weniger ideologische Einstellung gegenüber seinem Projekt an den Tag legt und quasi nebenher eine weitgehend „normale“ Biographie mit Frau und Kind sowie einem herkömmlichen Angestelltenverhältnis in gehobener Position vorweisen kann. Dr. Stallman und Linus Torvalds repräsentieren beispielhaft zwei unterschiedliche Entwicklungsmodelle Freier Software, welche am ehesten mit ideologisch versus technischer Motivation zu umschreiben sind[58].

Der Nerd

„Um eins allerdings machte ich mir Sorgen, als er größer wurde: Wie um alles in der Welt sollte er auf diese Weise jemals nette Mädchen kennen lernen?“

Anna Torvalds, Mutter von Linus Torvalds
(in: Torvalds/Diamond 2001, S. II)

Im angelsächsischen akademischen Umfeld hat sich besonders ein Begriff zur Umschreibung von technisch sehr gebildeten und überdurchschnittlich intelligenten, dabei aber sozial weniger kompetenten und sozial weniger eingebundenen, respektierten Individuen etabliert: Der „Nerd“. Die historischen Wurzeln des Begriffs gehen wahrscheinlich auf eine Comic-Figur aus dem Jahre 1950 zurück, auf zwei weitere mögliche Ursprünge sei hier nicht eingegangen. Der Nerd – in den meisten Fällen ist er männlich – entwickelt stereotypisch ein starkes Interesse an Wissenschaft, Science Fiction und Computern und gilt als unbeholfen in sozialen Interaktionen. In der Regel ist der Nerd auf seinem Interessengebiet überdurchschnittlich kompetent und tauscht sich auch mit anderen Nerds darüber aus, tritt aber kaum in Verbindung zu Nicht-Nerds. Auch sein Bedürfnis nach Selbstbestätigung und Anerkennung zieht er aus der Interaktion mit anderen Nerds. Der Begriff wird heute allerdings sehr schwammig für alle möglichen Technik-Interessierten verwendet und hat einiges seiner ehemals negativen Konnotation verloren. So schmückt sich beispielsweise eine der populären englischsprachigen Nachrichtenquellen für Themen um Computer und Internet – Slashdot[59] – mit dem Untertitel: „News for Nerds. Stuff that matters.“

Dem Autor ist keine empirische Studie bekannt, die den tatsächlichen Anteil der Nerds an und in Freier Software-Entwicklung zu bestimmen versucht – nicht zuletzt auch aufgrund der problematischen Definition eines Nerds. Aus den persönlichen Erfahrungen des Autors in diesem Bereich und durch die Interviews sowie Beobachtungen des Umfelds der Interviews – dem Linux Tag 2004, der größten europäischen Messe zum Thema Freie Software – unterstelle ich einen tatsächlich überproportional hohen Anteil von „Nerds“ – allerdings anhand des Bildes des „aufgeweichten“ Stereotypes und nicht des Prototypen eines Nerds. Eventuell kann man von einer teilweise „extremen Fokusierung“ des Interesses eines Großteils der Akteure in der Freien Software-Produktion sprechen, welche sich unter Umständen auch auf Kosten anderer Interessen oder der sozialen Interaktion niederschlagen kann. Im Rahmen des FLOSS-Projektes[60] wurden unter anderem dieser Aspekt empirisch erhoben und man kam zu einem ähnlichen Ergebnis: „However, the often drawn picture of the computer-sticking student, spending hours and hours for developing OS/FS and participating in the community’s communications, holds not true.“ (Ghosh et al. 2002, S. 26)

Die folgenden Abschnitte (Kapitel 3.2.3 bis 3.2.6) umreißen stichwortartig die Frage nach dem „Wer“ in Freier Software.

Bildung und beruflicher Hintergrund

Einen universitären Bildungsgrad besitzen 70 % der FLOSS-Stichprobe (Ghosh et al. 2002, S. 13). Über 2/3 der Befragten gehen zudem einer geregelten, bezahlten Tätigkeit im IT-Sektor nach oder sind an Universitäten im IT-Bereich beschäftigt (ebd., S. 210). Arbeitslosigkeit scheint kein Thema unter Freien Software-Entwicklern zu sein (ebd.).

Was die Mobilität der Freien Software-Entwickler betrifft, so sind Aussagen dazu mit Vorsicht zu genießen. Die teilweise sehr lose Zusammenarbeit über das Internet läßt sich oft lediglich anhand der Email-Adressen der Beteiligten rekonstruieren. Da es aber keine zwingende Verbindung zwischen der verwendeten Email-Toplevel-Domain – beispielsweise *.de oder *.com – und der tatsächlichen Heimat oder Arbeitsstätte eines Entwicklers gibt, lassen sich hierzu lediglich spekulative Aussagen formulieren. Allerdings zeigen verschiedene Studien, daß Europa sowie Nord-Amerika den Markt der Freien Software-Entwicklung dominieren.

Frauen

Provokativ formuliert kommt das weibliche Geschlecht[61] in der Freien Software-Entwicklung nicht vor[62]. Die empirisch ermittelten Zahlen variieren zwar, jedoch bewegen sie sich durchweg auf einem sehr niedrigen Teilnahme-Niveau. Tuomi berichtet in seiner Studie über die →Kontributoren des Linux-Kernels aus dem Jahre 2004, daß „[m]ost spectaculary, 99,6 percent of Linux developers mentioned in the Credit file are male.“ (Tuomi 2004, S. 15). Dies stellt allerdings aufgrund der Methodik Tuomi's einen Extremwert dar, da er sich ausschließlich einer Auswertung des Kernel-Codes widmet, also dem Bereich der Betriebssystem-Technologie und somit andere Software-Genres ausblendet. Dabei ist beispielsweise nicht nur an die Programmierung konkreter →Anwendungssoftware zu denken, sondern auch an eine Reihe ergänzender Tätigkeiten wie die Pflege von Dokumentation, Kommunikation mit Dritten oder Übersetzung – die Liste ließe sich fast unendlich fortsetzten. Eine in diesem Sinne „breitere“ Perspektive nimmt der FLOSS-Bericht[63] ein, der 2784 Freie- und Open-Source-Software-Entwickler umfaßt. Diese Stichprobe hat einen Frauen-Anteil von immerhin 1,1 % (Ghosh et al. 2002, S. 9). Begründungen dafür werden in beiden Studien entweder überhaupt nicht – wie in dem FLOSS-Bericht – oder nur sehr rudimentär mit Verweis auf eventuell unterschiedliche Opportunitätskosten zwischen Männern und Frauen geliefert – wie dies bei Tuomi der Fall ist. Die Darstellung, daß „women and men exist in two different economic spheres, or that women are paid much better for commercial software development than men.“ (Tuomi 2004, S. 18) erscheint aber etwas weit hergeholt.

Es bleibt hervorzuheben, daß diese Thematik zunehmend Aufmerksamkeit auf sich zieht und sich Interessengemeinschaften diesbezüglich gebildet haben. Insbesondere ein Projekt wie →Debian mit einem gesellschaftlichen, sozialen Anspruch[64] muß sich in diesem Punkt engagieren – was seit August 2004 auch öffentlich in Form eines Debian-Unterprojekts „Debian Women“[65] geschieht. Neu sind diese „Special-Interest“-Gruppen im IT-Sektor nicht, jedoch für den Bereich der Freien Software. So besitzt keine anderer GNU/Linux-Distribution eine derart institutionalisierte Vertretung des weiblichen Geschlechts wie Debian. Etwas länger – seit 1999 – besteht eine Distributions-unabhängige Vertretung in Form von „Linuxchix“[66] für den gesamten Freien Software-Bereich[67]. Aufschlußreich zum Einen bezüglich des bisher geringen Engagements von Frauen in Freier Software – und im IT-Sektor allgemein[68] – und zum Anderen über die Ursachen der Gründung einer eigenen Interessenvertretung und Anlaufstelle ist, daß Frauen in diesem Bereich anscheinend massiv ausgegrenzt und nicht ernst genommen werden. Raven Alder - Software-Entwicklerin, Netzwerk- und UNIX-Administratorin – hat ihre Erfahrungen hierzu sehr plastisch zum Ausdruck gebracht:

„Who is Starla Pureheart? She's a hacker among many others. She frequents security conventions and hangs out on mailing lists and online communities. But apparently Starla has committed the heinous crime of having breasts as well as a couple of shell accounts. Clearly, this renders all of her technical accomplishments completely irrelevant. I mean, she's a *girl*. Everyone knows that girls aren't really [Hervorh. i. Orig.; Anm. d. Verf.] hackers.“ (Raven 2004, S. 1) [„Starla Purehart“ eventuell ein Phantasiename; vgl. Namen des Curriculum Vitae: „Alder Raven“[69]]

Partnerschaft und Familie

Aus den Daten des FLOSS-Berichts – der einzigen Quelle zu diesem Aspekt – geht hervor, daß das Verhältnis von Freien Software-Entwicklern, die als Single oder in Partnerschaft leben, annähernd zwei zu drei beträgt. Partnerschaft schließt auch nicht-eheliche Partnerschaften mit

ein. Nur 17 % der Stichprobe von Freien Software-Entwickler haben Kinder (Ghosh et al. 2002, S. 208).

Jugend

Eine sehr große Rolle in der Produktion von Freier Software spielen junge Menschen, die sich oft noch in der Ausbildung befinden. Die Autoren des FLOSS-Studie berichten von einem Anteil an Studenten in der Freien Software-Entwicklung von 21 % (Ghosh et al. 2002, S. 9ff). Damit stellen Studenten nach den hauptberuflichen Freien Software-Entwicklern die zweitgrößte Gruppe (ebd.). Bezüglich der Altersstruktur der Freien Software-Entwickler wird von einem Durchschnittsalter von 27 Jahren berichtet, wobei der Median bei 26 Jahren liegt. Das gemittelte „Eintrittsalter“ in die Freie Software-Entwicklung wird mit knapp 23 Jahren angegeben (ebd.).

Die Interviews bestätigen dieses Bild. Marco Schulze formulierte seinen Einstieg in die Programmierung folgendermaßen:

„Die größte Motivation ist Spaß und Neugier. Die meisten Software-Entwickler fangen als Kinder an, Software zu entwickeln. Das war bei mir so, meinen ersten Rechner hatte ich zu meinem 15. oder 16. Geburtstag geschenkt bekommen. Das war relativ spät [...] und sofort angefangen Software zu entwickeln.“ (Schulze 2004, S. 7)

Eine ähnliche Einschätzung treffen Werner Koch, der ebenfalls von einem Eintrittsalter von „15 oder 16“ (Koch 2004, S. 1) spricht, und Martin Michlmayr, der mit „13 oder 14“ (Michlmayr 2004, S. 13) über seinen Bruder einen ersten Zugang zu einem Rechner einer Universität hatte. Allerdings spiegeln die in den Interviews genannten Daten nicht das Eintrittsalter in die Entwicklung von Freier Software wieder, sondern lediglich den ersten Kontakt mit Computern und eigener Programmierung. Dabei kann oft nicht von einem Datum im Sinne eines konkret definierbaren Zeitpunktes gesprochen werden, da sich das Engagement in Freier Software eher beiläufig entwickelte und mit der Zeit immer mehr Platz in der Software-Entwicklung der Befragten einnahm.

Untersuchungen zu Engagement und Motivation

Literatur und Methodik

Die Literatur zur vorliegenden Thematik ist auf der einen Seite sehr vielfältig, auf der anderen Seite jedoch noch nicht in dem Maße „ausgereift“, wie man es von der Literatur gängiger soziologischer Themen kennt. Auffallend – aber nicht überraschend – ist das Fehlen von „Basisliteratur“ zum Thema. Diesem Mangel steht allerdings ein fast unüberschaubar großer Pool von Artikeln in online publizierter Form gegenüber. Artikel, die hauptsächlich aus dem akademischen, universitären Bereich stammen. Aber auch diverse Forschungsinstitute, Organisationswissenschaftler aus dem ökonomischen Bereich und inzwischen die Lobby der Kulturschaffenden und Künstler steuern einen Teil der Literatur bei. Die Nähe zum akademischen Bereich ist eine direkte Folge der Ursprünge und der Verwendung von Freier Software und Open Source Software: Ihre „Geburtsstätte und ihre Kindheit“ war an universitäre oder universitätsnahe Einrichtungen gebunden, auch die hauptsächlich frühe Förderung wurde von der Schirmherrschaft der Universität übernommen. Und noch heute sind universitäre Einrichtungen eine der stärksten Lobbys für Freie Software[70].

Auffällig ist die erschlagende Dominanz von wenigen Artikeln aus der Pionierzeit Freier Software[71]. Die aktuelle Perspektive auf diese Artikel – wir würden sie heute als „Klassiker“ bezeichnen – würdigt weniger die Qualität, sondern vielmehr die Leistung der entsprechenden Autoren, das Phänomen Freie Software festgehalten und einer wissenschaftlichen Analyse unterzogen zu haben, sowie die Möglichkeit geschaffen zu haben, das Phänomen weiteren Untersuchungen zugänglich zu machen. Die Euphorie, welche diese ersten Texte auslöste, ist heute verflogen. Es mehrt sich Kritik selbst an zentralen Punkten[72]. Aber dieser Diskurs ist trotz allem noch verhältnismäßig jung – zu jung um bisher ein „Standardwerk“[73] oder eine Lehrbuchmeinung hervorgebracht zu haben.

Um diese zum gegenwärtigen Zeitpunkt noch kleine Basis weiter zu festigen, wird auf die persönliche Erfahrung des Autors, eines langjährigen Anwenders und Befürworters von Freier Software, sowie insbesondere die qualitative Untersuchung, welche im folgenden Kapitel kurz vorgestellt wird, zurückgegriffen.

Übersicht Interviews

Um diese Basis zu verbreitern, wurden vier qualitative Interviews anhand eines Gesprächsleitfadens mit teilweise international bedeutenden Akteuren Freier Software durchgeführt. Die Interviewdauer betrug zwischen einer halben Stunde – im Fall von Harald Welte – und knapp über eine Stunde – bei Marco Schulze. Die beiden verbleibenden Interviews dauerten je knapp 45 Minuten. Die Interviewpartner waren im Einzelnen:

Werner Koch

- Autor von GnuPG, Vize-Kanzler der Free Software Foundation Europe
- http://www.fsfeurope.org/about/team.en.html; http://www.guug.de/~Werner.Koch/
- Aufzeichnung: 25.06.2004; 16:00 – 17:00 Uhr; Linux Tag 2004, Karlsruhe

Martin Michlmayr

- Debian Projekt-Leiter, Entwickler, Forschung
- http://www.cyrius.com/cv/index.html; http://www.debian.org/vote/2004/platforms/tbm.en.html
- Aufzeichnung: 25.06.2004; 14:30 – 15:20 Uhr; Linux Tag 2004, Karlsruhe

Marco Schulze

- Geschäftsführer, Projektleitung, Programmierung
- http://wiki.nightlabs.de/en/Main_Page
- Aufzeichnung: Dienstag, 15.08.2004; 18:00 – 21:00 Uhr; Büroräume der NightLabs GmbH, Rehlingstr. 6, Freiburg

Harald Welte

- Programmierer; „Active Core Member“ des netfilter/iptables-Projekts und dessen offizieller Vertreter

- http://www.netfilter.org/about.html#coreteam
- Aufzeichnung: 26.06.2004; 14:00 – 14:40 Uhr; Linux Tag 2004, Karlsruhe

Pressesprecher Microsoft Deutschland GmbH

- Namentliche Nennung wurde durch den Interviewpartner untersagt, ebenso die Aufzeichnung des Gesprächs
- http://www.microsoft.com/germany/ms/unternehmensinformationen/gmbh_profil/
- Stattgefunden am: 25.06.2004; 15:30 – 15:50 Uhr; Linux Tag 2004, Karlsruhe

Transkribierte Fassungen sämtlicher Interviews sind unter der URL http://www.tombreit.de/vers2/sites/uni/ jederzeit abrufbar. Unter der selben URL ist auch eine hypertext-fähige Version der vorliegenden Arbeit zugänglich.

Der Gesprächsleitfaden sollte den Interviewpartnern bewußt den nötigen Freiraum gewähren, daß sie ihre persönliche Geschichte von Freier Software vorbringen konnten, anstatt sie mit konkreten Fragen auf bestimmte Aussagen zu lenken. Im Regelfall wurde das Gespräch über erste persönliche Beziehungen mit dem Computer oder Programmierung über die Entwicklung dieser Beziehungen auf aktuelle Fragestellungen gelenkt. Es wurde vermieden, konkrete Motivationen abzufragen, da diese sich aus den Ausführungen der Interviewpartner meist ergaben. War dies der Fall, wurde explizit nachgefragt, um eine Bestätigung bestimmter Motivationen zu erhalten.

Diese offene Strategie erwies sich als sehr hilfreich. Es wurde in jedem Fall eine enge Verzahnung der eigenen persönlichen Geschichte und der Erfahrung mit Freier Software zum Ausdruck gebracht. Ein Wechselspiel, welches exemplarisch unterschiedlichste Motivationen beinhaltet und sich mit einer Reihe von früheren Befunden[74] deckt. Die Behandlung der einzelnen Motivationen findet in den Kapiteln 5.1 bis 5.7 statt. Hingewiesen sei auf die – je nach Interviewpartner – sehr persönliche Batterie von Motivationen sowie die Verschiebungen der Gewichtung einzelner Motivationen im zeitlichen Verlauf.

Es war bei jedem Interviewpartner – mit der Ausnahme des Microsoft-Pressesprechers – eine hervorragende Gesprächsbereitschaft gegeben. Sämtliche Interviews kamen mit geringen Anreizen seitens des Interviewers aus, auch wenn dadurch oft die Vorstrukturierung durch den Gesprächsleitfaden gesprengt wurde.

Auf eine Kurz-Charakteristik der einzelnen Interviewpartner wird an dieser Stelle verzichtet, da bei der Heranziehung der Interviews zur Darstellung der Motivationen auf die Persönlichkeiten näher eingegangen wird und ihre offizielle Funktion innerhalb der Freien Software-Community jeweils angegeben ist (s.o.). Die Interviews fließen thematisch an den entsprechenden Stellen in die Arbeit ein.

Das vielleicht interessanteste Ergebnis der Interviews ist – um es dem Hauptteil vorwegzunehmen – daß sie die im Vorfeld eröffneten Dimensionen von Motivation in Freier Software – Reputation, Altruismus, Community, Ideologie und Spaß (siehe Kapitel 5) bestätigen.

Der „Spezialfall“ des Microsoft-Interviews verbietet leider eine genauere Analyse. Hier wurde die Aufzeichnung sowie die namentliche Nennung des Interviewpartners untersagt. Das Inter-

view fand trotzdem in Form eines informellen Gesprächs, welches nicht die Meinung der Microsoft Deutschland GmbH darstellt, statt. Bestimmte Aussagen wurden so gut wie möglich im Nachhinein aufgezeichnet, können aber aufgrund methodischer Bedenken nicht in diese Arbeit einfließen[75].

Wegbereiter eines Raumes für Freie Software

Maslow und die Bedürfnispyramide

Die Diskussion von Motivation und Engagement in Freier Software findet nicht auf der Ebene einer Analyse im Sinne des „technischen“[76] Motivationsbegriffes statt, sondern impliziert die Hinzunahme der sozialen Einbindung des betreffenden Individuums. Das Phänomen der Freien Software-Entwicklung ist an gewisse gesellschaftliche Voraussetzungen gebunden und ist Ausdruck von Biographien, die zunehmend von Selbstentfaltung, Selbstdarstellung und einer einer „Freizeitkultur“ geprägt sind.

Legt man die Maslow'sche Bedürfnispyramide (Maslow 1978, S. 153ff) zugrunde, so sind in allen Ausführungen in dieser Arbeit die Grundbedürfnisse – nach Maslow die Stufen eins und zwei, respektive physiologische Bedürfnisse und Sicherheitsbedürfnisse – der Bedürfnispyramide anderweitig gedeckt. Die Beschäftigung mit Freier Software allerdings kann auf jeder der drei höheren Stufen stattfinden: Auf der nächsten, der dritten Stufe der sogenannten Zugehörigkeits- und Liebesbedürfnisse (siehe Kapitel 5.3), auf der vierten Stufe der Wertschätzungs- und Geltungsbedürfnisse (siehe Kapitel 5.1) und auf der höchsten Stufe des Bedürfnisses nach Selbstverwirklichung. Trotz aller Kritik an Maslow's Konzept, welches inbesondere die letzte Stufe, die Selbstverwirklichung[77], thematisiert, zeigt die breite Zuordenbarkeit von Motivationen in Bezug auf Freie Software, sowie daß wir es nicht mit einem rein Hedonismus-orientierten Phänomen zu tun haben, wie dies beispielsweise von jüngsten Untersuchungen zum Flow-Konzept[78] vorgetragen wird.

Die Analyse der Interviews sowie weitere Quellen belegen das breite und oft tief in das persönliche Schicksal eingreifende Feld der Beschäftigung mit Freier Software. Für Maslow bleibt zu erwähnen, daß auch er trotz der relativen Statik seines Pyramidenmodells explizit auf das Zusammenwirken verschiedener Motivationen hinweist. Dieser Umstand, den Maslow „Multiple Motivationen“ (Maslow 1978, S. 100) nennt, deckt sich in der Konsequenz mit den Erkenntnissen Herzbergs[79].

Inglehart und der Wertewandel

Auch Ronald Inglehart knüpft in seinem Werk „Culture Shift“ (Vgl.: Inglehart 1990) an die Maslow'sche Bedürfnispyramide an, wenn er seine Postmaterialismus-Theorie postuliert. Die von Inglehart auf breiter empirischer Basis erhobenen Daten zur Wandlung von Werten in hauptsächlich westlichen „Post-World-War-II“-Gesellschaften interpretiert er derart, daß sich moderne Gesellschaften auf einer Materialismus – Post-Materialismus-Dimension einordnen lassen. Beschreibt Maslow seine Bedürfnisspyramide noch wertfrei, unterstellt also dem Erreichen der einzelnen Stufen Automatismen, welche greifen sobald die jeweiligen notwendigen Bedingungen der vorigen Stufe erfüllt sind, so argumentiert Inglehart auf einem weniger universell anwendbaren Niveau. Er geht einen Schritt weiter und versucht eine spezielle, westliche Nachkriegs- oder Aufschwungs-Wertewandlung zu erklären. Inglehart konzeptualisiert das Maslow'sche Schema neu und generiert eine 12-Item-Skala mit zugehörigen Bedürfnissen auf der Materialismus-Dimension[80]. Seine Erkenntnisse zum Postmaterialismus decken sich in

vielen Fällen mit Erklärungen in Bezug auf individuelles Engagement in Freier Software: Sobald bestimmte notwendige Bedürfnisse ausreichend gedeckt sind, kann sich das Individuum um sogenannte „höhere“ Bedürfnisse kümmern. In der Geschichte von Freier Software, wie auch bei den vorliegenden Interviews, ist deutlich hervorzuheben, daß Freie Software-Entwicklung und Engagement in einem Raum von „Luxus“ im Sinne von Überfluß stattfindet. Dieser „Luxus“ kann hier verstanden werden als ein Luxus an Zeit, wie ihn der Finne Linus Torvalds (vgl.: Kapitel 2.1.5) genoß, kann als ein „Luxus“ an Know-How und Technik verstanden werden, wie ihn Richard Stallman (vgl.: Kapitel 2.1.3) an den →AI-Labs vorfand, oder aber auch als Überfluß an finanziellen Mitteln, die die entsprechende Institution, Einzelperson oder das Unternehmen nutzt, um Freier Software „unter die Arme zu greifen“[81]. Der Leiter eines der größten Freien Software-Projekte[82], Martin Michlmayr, formuliert diesen Umstand mit den Worten:

„Vor ein paar hundert Jahren hat einfach jeder von früh bis spät arbeiten müssen, um zu überleben. Und jetzt braucht man nur noch ein paar Stunden im Monat zu arbeiten, um eine Wohnung zu haben, um zu essen. Man hat soviel Freizeit, wo du Sachen machen kannst. Ein paar Leute werden sich sicher vor den Fernseher setzen und abschalten, aber manche – und ich glaube das ist die Mehrzahl – die wollen auch was Kreatives machen. Und ich denke, daß es dann auch viel mehr Freiwillige gibt, die etwas machen wollen – nicht nur in der Software, auch in allen anderen Bereichen.“ (Michlmayr 2004, S. 13)

Ergänzend erwähnt Harald Welte als Voraussetzung, daß „das ist ja eben – ich muß nur diese Sprache sprechen – C [die Programmiersprache in der der Linux-Kernel geschrieben wurde; Anm. d. Verf.] in diesem Fall – und Zeit haben, und dann kann ich im Prinzip egal was“ (Welte 2004, S. 5).

Nun ist es Inglehart's Anliegen nicht, die wirtschaftliche Prosperität zu ergründen und zu beschreiben – seine Forschungen finden vielmehr in dem zusätzlichen Raum statt, den dieser „Luxus“ den Menschen bereitet, in Inglehart's Semantik ausgedrückt durch die „höheren“ sechs Items „More to say in government“, „More to say on job, community“, „Less impersonal society“, „Free speech“, „Beautiful cities/Nature“ und „Ideas count“ (Inglehart 1999, S. 134). Diese Inglehart'schen postmaterialistischen Werteorientierungen finden sich auch in der Ethik der Freien Software-Entwickler wieder – wenn auch dort auf andere Weise formuliert:

„Überhaupt habe ich unglaublichen Respekt vor Leuten, die irgendwelche Sachen, irgendwelche fertigen Lösungen nehmen, zerlegen, um etwas komplett anderes daraus zu machen. Das ist diese Hacker-Mentalität: kreativer Umgang mit irgendwas.“ (Welte 2004, S. 6)

Betrachtet man die auffälligste Übereinstimmung: „Ideas count“ (Inglehart 1990,S. 134), so ist dies die Maxime der Freien Software-Entwicklung. Was zählt ist die Innovation und die technische Umsetzung. Auf dieser Ebene mißt man sich im Software-Bereich, und insbesondere innerhalb der Freien Software-Gemeinde ist die Qualität und Originalität des Codes das einzige Wettbewerbsmerkmal des jeweiligen „Produktes“:„Es gibt für keinen Programmierer, glaub ich, ein höheres Ziel als möglichst guten Quellcode zu schreiben, möglichst gute Software zu schreiben. Genauso wie es für einen Komponisten einfach ein Anreiz ist, schöne Musik zu komponieren.“ (Schulze 2004, S. 16)

Kategorisierung und Untersuchung ausgewählter Motivationen

Bisher haben wir uns in Kapitel 2 mit dem historischen Kontext von Freier Software-Entwicklung vertraut gemacht sowie in Kapitel 3 und 4 wichtige Umgebungsvariablen eines Engagements in Freier Software kennen gelernt. Nun sollen die – vor allem aus soziologischer Perspektive – wichtigsten Motivationsfaktoren näher betrachtet werden. Das entscheidende Schlüsselwort bei einer solchen Zusammenstellung ist die Heterogenität[83]: die Heterogenität der Akteure, die Heterogenität der Motivationen sowie die Heterogenität der Erkenntnisse der vorhanden Literatur. Eindeutige Kategorisierungen fallen oft schwer, da sich die unterschiedlichen Motivationsfaktoren gegenseitig beeinflussen, sich im zeitlichen Verlauf die Bewertungen verschieben und aus einer rein empirischen Perspektive sich Motivationsfaktoren nicht direkt beobachten lassen, sondern lediglich ex post deren Konsequenzen sich interpretieren lassen. Auch der Interviewpartner Marco Schulze formuliert diesen Umstand: „Welcher Punkt da überwiegt [welche einzelnen Motivationen; Anm. d. Verf.] ist da schwer zu sagen, und sicherlich von Person zu Person unterschiedlich. Und sicherlich innerhalb einer Person von Zeit zu Zeit unterschiedlich.“ (Schulze 2004, S. 16)

Eine sehr häufig verwendete Möglichkeit der Einordnung von Motivationen basiert auf der Unterscheidung von „extrinsisch“ und „intrinsisch“. Dieses Schemata kann und wird sowohl in der allgemeinen psychologischen Literatur[84], wie auch für den speziellen Fall der Freien Software verwendet[85]. Der Vorteil liegt in dem theoretischen Fundament der Motivations-Psychologie und der in den meisten Fällen sehr exakten Zuordenbarkeit der entsprechenden Motivationsfaktoren zu einer der beiden Kategorien. Da diese beiden Begriffe – extrinsisch und intrinsisch – auch hier von zentraler Bedeutung sind, seien sie kurz umrissen: extrinsisch motiviert beschreibt einen Zustand, der von außen angeregt wird, der nicht aus eigenem, inneren Antrieb angestrebt wird. Nah verwandt mit extrinsischer Motivation sind Begriffe wie „äußerer Zwang“ und „Strafe“. Da jede Motivation letztlich Prozesse im jeweiligen Individuum darstellt, und sämtliche resultierende Handlungen und Verhaltensweisen von diesem Individuum ausgehen – also unabhängig von der Kategorisierung nach intrinsisch oder extrinsisch sind – ist für das Verständnis der extrinsischen Motivation eventuell das Bild des „Mittels zum Zweck“ hilfreich: Eine extrinsische Motivation muß demnach anliegen, wenn die Tätigkeit an sich nicht „reizvoll“ genug ist und demnach nicht ausgeführt werden würde. Ist lediglich die Konsequenz einer Handlung oder eines Verhaltens erwünscht, aber nicht die mit dem Erreichen desselben zusammenhängenden Handlungen, so müssen weitere Reize, weitere Motivationen gesetzt werden. Dies entspricht der extrinsischen Motivation. Sie ist nötig, um ein bestimmtes Ziel zu erreichen, wenn der Prozeß dieses Erreichens negativ oder un-motivierend belegt ist. Als Beispiel sei das klassische abhängige Arbeitsverhältnis genannt, welches durch die Zahlung von Lohn einen „äußeren“ Anreiz schafft, bestimmte Tätigkeiten zu vollbringen, auch wenn diese nicht im „persönlichen“ Interesse der entsprechenden Person stehen. Ebenso ist die Weigerung, auf diesen äußeren Anreiz zu reagieren, in diesem Fall beispielsweise Leistungszurückhaltung am Arbeitsplatz, mit Sanktionen bedacht, welche von einfachem Tadel bis zur Kündigung reichen können.

Dem gegenüber steht die intrinsische Motivation. Hier liegt die Erreichung eines bestimmten Zustandes im primären, persönlichen Interesse. Das Erreichen dieses Zustandes erfolgt aus innerem Antrieb. Im Vergleich zur extrinsischen Motivation liegt „der Reiz“ in der Sache, der Tätigkeit selbst. Ein Beispiel ist das Hobby, das in sich genug Spaß und Befriedigung für den

Ausführenden birgt, er benötigt keine weiteren, externen Motivationen um seiner Lieblingstätigkeit nachzugehen.

Diese analytische Trennung von extrinsischer und intrinsischer Motivation funktioniert allerdings nur bei der isolierten Betrachtung der einzelnen Motivatoren[86].

Eine exakte Einordnung auf Ebene des Individuums funktioniert nach diesem Schema allerdings nicht. Die erwähnte Heterogenität und Variabilität der Zusammensetzung von verschiedenen Motivationen in ein und derselben Person ermöglicht höchstens noch eine Zuordnung im Sinne von „eher extrinsisch“ vs. „eher intrinsisch“ motiviert. Auch ist eine derartige Zuordnung nicht über die Zeit und Individuum stabil.

Es stehen noch weitere Zuodnungsmöglichkeiten zur Verfügung, welche aber alle den selben Einschränkungen unterworfen sind. So könnte man auf der Ebene des Individuums von „pragmatisch“ versus „ideologisch“ Motivierten sprechen, doch auch dann ist man mit dem Problem konfrontiert, daß diese Zuschreibungen über die Zeit nicht stabil sind[87].

Dieser Einwand gegen eine isolierte Betrachtung einzelner Motivationen ist nicht nur hypothetischer Natur, er manifestiert sich bei Richard Stallman[88] ebenso wie im Interview mit Harald Welte[89]. Diese Beschränkung ist bei der Kategorisierung auf Ebene des Individuums schwer zu umgehen.

Auch Moon und Sproul versuchen dieses verteilte Software-Entwicklungsmodell aus drei „beispielhaften“ Perspektiven zu beschreiben: Der Linus Torvalds-Methode, der →Hacker-Ethik und der Community (Moon/Sproul 2000, S. 3-9). Andere Autoren wie Osterloh et al. versuchen das Problem durch die Identifikation von Umwelteinflüssen zu lösen, und kategorisieren nach drei Aspekten: „[...] namely the motivational, situational and institutional factor.“ (Osterloh et al. 2002, S. 17)

Im Rahmen der vorliegenden Arbeit wird eine weitere Kategorisierung verwendet, die nach dem Grad der „Sozialität“ der einzelnen Motivationen fragt. Betrachtet man einige mögliche Motivationen wie Reputation, Spaß, Ideologie oder den technischen Mangel – nähere Ausführungen zu den einzelnen Motivationen in den Kapiteln 5.1 bis 5.7 – so fällt auf, daß sie sich in der Einbeziehung Dritter unterscheiden, eine Unterscheidung, die aus der soziologischen Perspektive besonders relevant ist.

Sollte eine Motivation – beispielsweise die Behebung eines Mangels – isoliert und einmalig auftreten, ist sie für diese Arbeit so gut wie ohne Bedeutung – abgesehen davon, daß der Mangel meist eine notwendige Bedingung für das eigene Aktiv-werden in Freier Software ist. Zu beachten ist allerdings, daß dieses „Aktiv-werden“ für viele Software-Entwickler oftmals das „Sprungbrett“ in Freie Software war und ist, und somit eben jener erwähnten Variabilität über die Zeit entspricht und in ihrer Konsequenz – hier: dem Hinzukommen von weiteren Motivationen – soziologisch bedeutsam werden kann. Auch sei auf den hohen Stellenwert dieser Motivation des Mangels hingewiesen. In jedem Interview und in allen wichtigen Publikationen zu der Thematik finden sich Belege für die initiale Motivation des Mangels (siehe auch Kapitel 5.7).

Für diese Arbeit sind somit jene Motivationen von Interesse, für welche als notwendige Bedingung ein Dritter involviert sein muß. Man muß von einem „wir“ oder von der Interaktion innerhalb einer Gruppe sprechen können, um dieser Bedingung zu genügen. Nach dieser Kategorisierung werden die Motivationen Reputation, Altruismus, Community sowie Ideologie einer

tieferen Analyse unterzogen, wogegen auf Hedonismus, Mangel und Weiterbildung als nicht notwendig soziale Motivationen lediglich ergänzend eingegangen wird.

Es sei an dieser Stelle betont, daß es sich bei den für diese Arbeit sekundären Motivationen keineswegs um weniger bedeutsame Motivationen handelt – ganz im Gegenteil, leisten sie doch die größte Varianzaufklärung – jedoch können sie hier nur als ergänzende, komplementäre Aspekte behandelt werden, da sie der soziologischen Fragestellung nicht in genügendem Maße gerecht werden. Ihre Existenz hilft aber auch bei der Untersuchung von Reputation, Altruismus, Community sowie Ideologie als Motivationen für ein Engagement in Freier Software: Die schon hervorgehobene Heterogenität der einzelnen Motivationen und deren teilweise gleichzeitiges Auftreten mit unterschiedlicher Gewichtung impliziert auch das Auftreten der hier als sekundär eingestuften Motivationen. Diese können somit einen zusätzlichen Erklärbeitrag leisten. Insbesondere muß man sich ihrer bewußt sein, denn sie können Verhalten erklären, welches nicht in den „sozialen“ Kategorien zu fassen ist. Man kann zwar beispielsweise von einem Reputations-orientierten Individuum und seinem daraus resultierenden Beitrag zu Freier Software sprechen, doch müßte dieser Beitrag bei Nicht-Erfüllung des Reputationsbedürfnisses gegen Null gehen. Dem ist aber in der Regel nicht so, denn weitere Motivationen halten weiterhin ein bestimmtes Beitrags-Niveau. Anders ausgedrückt: Auch wenn man die ersehnte Anerkennung nicht bekommt, macht die Tätigkeit an sich eventuell noch beispielsweise genug Spaß, um sie fortzuführen. Ähnlich verhält es sich mit dem technischen Mangel (siehe Kapitel 5.6) und der Weiterbildung (siehe Kapitel 5.7).

Reputation

„You do not become a hacker by calling yourself a hacker – you become a hacker when other hackers call you a hacker [Hervorh. i. Orig.; Anm. d. Verf.].“

(Raymond 2001b, S. 94)

Das Streben nach Anerkennung ist sicherlich einer der wichtigsten Ansätze zur Erklärung des Phänomens Freier Software. Reputation ist zudem der Qualitätsmaßstab für Freie Software, wie es auch für andere Bereiche – etwa der Kunst – gilt, die nicht der Knappheit unterworfen sind[90]. Das Bedürfnis nach Respekt, nach Anerkennung der eigenen Persönlichkeit und des eigenen Schaffens sind ein „zwingend mit dem menschlichen Selbstbewußtsein verknüpftes Problem.“ (Popitz 1987, S. 633). Man kann den Bereich Reputation aufteilen nach dem Bedürfnis innerhalb einer bestimmten Gruppe, einem „Kreis von Gleichgesinnten“ Anerkennung zu erlangen, sowie in das Streben nach „globaler“ Anerkennung, einer Anerkennung, welche über die primären, direkten Bezugsgruppen hinausgeht. Das Streben nach Reputation erfolgt nicht Kontext-frei, es kann bestimmte nachgelagerte Ziele verfolgen. Lerner und Tirole heben die Reputation über alle anderen, möglichen Motivationsfaktoren und betonen die externe Signalwirkung einer Reputation als „guter Programmierer“, die über die Freie Software-Community hinaus strahlt und somit Arbeits- und Karrierechancen erhöhen kann:

„What motivates programmers? Theory. A programmer participates in a a project, wheter commercial or open source, only if she derives a net benefit (broadly defined) from engaging in the activity. The net benefit is equal to the immediate payoff (current benefit minus current cost) plus the delayed payoff [Hervorh. i. Orig; Anm. d. Verf.].“ (Lerner/Tirole 2000, S. 14)

Dieser „delayed payoff“ [dt.: verzögerte Auszahlung] im Sinne von erhöhten „career concern incentive[s]“ (ebd., S. 14) [dt.: Karrierechancen] außerhalb der Freien Software-Community

konnte aber bisher nicht empirisch nachgewiesen beziehungsweise von Dritten bestätigt werden (Vgl.: Lakhani 2003, S. 6).

„Climbing on the Shoulders of Giants“

„To me programming is more than an important practical art. It is also a gigantic undertaking in the foundations of knowledge.“
(Grace Hopper in: DiBona et al. 1999, S. 7)

Robert K. Merton prägte diesen Ausspruch in der Soziologie, als er die Geschichte dieses Aphorismuses anhand der Geschichte der Wissenschaft und dem Selbstverständnis des Wissenschaftlers nachzeichnete[91]. Auch in der Literatur zu Freier Software finden wir Referenzen zu Merton's Klassiker. Osterloh et al. bauen ihre Argumentation zur Produktion von öffentlichen Gütern – hier Freier Software – und Motivation auf der Parallele zur wissenschaftlichen Produktion von Wissen auf (Vgl.: Osterloh et al. 2002, S. 14f).

Es lassen sich für das Freie Software-Modell zwei Schlüsse aus Merton's berühmten Spruch und der Verknüpfung mit der Produktion von Wissen in der Wissenschaft ableiten: Zum Einen die ähnliche Charakteristik von Wissen im akademischen Sinne und den Algorithmen, aus denen Software aufgebaut ist; zum Anderen die Ähnlichkeit in der Produktionsfunktion von Wissen und Software. Auch im Bereich der Freien Software kann auf einen großen Pool von „Wissen in Software“ zurückgegriffen werden, dieses Wissen weiter verwendet, kombiniert und transformiert werden um dadurch Neues zu schaffen. In Freier Software gilt es, abgeleitete Arbeiten und Ergebnisse wieder in diesen Pool zurückfließen zu lassen und damit den Pool zu vergrößern, oder – um in der Analogie der Riesen und Zwergen zu bleiben – den Riesen wachsen zu lassen, auf daß der nächste Zwerg auf seiner Schulter weiter sieht als seine Prädezessoren. Der individuelle Beitrag erfährt nach bestimmten Kriterien eine Gewichtung und Bewertung durch die anderen Nutzer dieses Pools. Weiter wird jeder Beitrag mit namentlicher Nennung fixiert. Dies geschieht im wissenschaftlichen Bereich durch den Citation Index[92]. Im Bereich der Freien Software wird diese Gewichtung und Bewertung von Beiträgen ebenfalls festgehalten, allerdings nicht in einem globalen Verzeichnis, sondern jeweils für die unmittelbare Umgebung eines Software-Projektes[93]. So kann, wenn man Merton's Ausspruch auf Freie Software überträgt, eher von einer Reihe verschiedener, unabhängiger Pools dieses in Software destillierten Wissens gesprochen werden, und jeder dieser Pools verfügt über seinen „eigenen Citation Index“ und weist den →Kontributoren durch die Gewichtung und Bewertung des Beitrags die entsprechende Anerkennung zu.

Die entscheidende Parallele ist die Ähnlichkeit der Produktionsfunktionen in der Wissenschaft und in Freier Software. Diese Produktionsfunktion ist aber auch in weiteren Bereichen zu finden, Grundvoraussetzung für die Anwendbarkeit einer solchen Produktionsfunktion ist dabei, daß es sich bei dem behandelten Gut um „Ideen“ handelt und diese damit keiner Knappheit im materiellem Sinne unterliegen. Damit ist die Abgrenzung der in dieser Arbeit verwendeten Produktionsfunktion gegenüber einer Güterwirtschaft gegeben. Beispielhaft seien weitere Ausprägungen dieses verteilten Produktionsmodells aufgeführt[94]:

- Wikipedia[95], eine verteilte Enzyklopädie, die es jedem Nutzer erlaubt, neue Einträge zu erstellen oder vorhandene Einträge zu verbessern und damit ein „öffentliches“ Nachschlagewerk schafft sowie die Weiterverbreitung und Modifikation der Inhalte ausdrücklich wünscht.

- Slashdot[96], verteilte Nachrichten, hauptsächlich IT-naher Themen, die von einer großen Gemeinde eingepflegt und kommentiert werden.
- Projekt Gutenberg[97], eine Sammlung von von Freiwilligen zusammengetragenen Büchern in elektronischem Format, welche durch ihr Alter in den Vereinigten Staaten als public domain, als öffentliches Gut, gelten.

Online-Communities und die Verführung der Entgrenzung

Die Frage nach der Reputation ist immer auch eine Frage nach dem Publikum. In der Regel „empfängt“ man Anerkennung und Ansehen von Dritten. Die in sich als erfüllend empfundene Tätigkeit sei hier ausgeklammert und auf Kapitel 5.5 verwiesen. Gehen wir weiter davon aus, daß es für jedes Individuum begrenzte Möglichkeiten gibt, für eine bestimmte Tätigkeit eine positive – sowie negative – Reputation zu erlangen, so verändern sich diese Möglichkeiten mit der Größe des Publikums. Aus der Perspektive des Publikums ist Aufmerksamkeit ein knappes Gut, und aus diesem Pool an Aufmerksamkeit kann der Freie Software-Entwickler sein Bedürfnis nach Anerkennung stillen. Demnach steigen mit der Größe des Pools auch die Chancen für den Entwickler, selber mit Aufmerksamkeit und eventuell mit positiver Reputation bedacht zu werden.

Das Entwicklungsmodell Freier Software ist aufgrund verschiedener Aspekte geradezu prädestiniert für ein optimales Verhältnis zwischen dieser öffentlichen Aufmerksamkeit und dem Schaffen des Entwicklers: Die offene Kommunikationsstruktur über das Internet via →IRC, →Mailing-Listen und →Foren, sowie die prinzipielle „Entgrenzung“ der möglichen erreichbaren Öffentlichkeit scheinen als ein Ansporn für das Engagement für Freie Software zu wirken. Wo sonst kann mit derart niedrigen Kosten eine derart große Öffentlichkeit erreicht werden? Dieser Gedanke der Entgrenzung des eigenen Schaffens wirft auch bezüglich einiger Aspekte des vermeintlich altruistischen Verhaltens[98] Fragen auf – ist doch manches „altruistisches“ Verhalten unter dieser Perspektive nicht mehr als die ego-zentrierte Suche nach Aufmerksamkeit in dem größten denkbaren Raum[99] mit den verhältnismäßig geringsten Kosten.

Wolf Lotter rechnet in BRAND EINS[100] die Entwicklung Freier Software gegen Tendenzen traditioneller Entwicklungsmodelle auf. Dabei kommt er unter anderem zu dem Ergebnis, daß „[d]ie Netzwerkökonomie [...] eine Wirtschaft der erweiterten Möglichkeiten [ist].“ (Lotter 2001, S. 30). Diese erweiterten Möglichkeiten beinhalten unter anderem, daß „[i]n addition to providing information and social benefits, online groups also provide opportunities for people to be visible beyond the boundaries of their local work or geographical community.“ (Lerner/Tirol 2000).

Einschränkungen der Erklärkraft der Reputation

Auch ist die „Einzelherrschaft“ der Reputation unter den Motivationen inzwischen mit Vorsicht zu genießen, da sie für eine Reihe von Phänomenen keine adäquate Erklärung bereitstellt. Maria Alessandra Rossi faßt eine Reihe von Einwänden prägnant zusammen:

„What the reputation argument does not make clear is why would a „top-notch programmer“ decide to initiate a project in the first place by rendering freely available on the web code that might have significant commercial value if taken private. [...] [I]f reputation were the primary driver of contributions, we would probably observe significantly more direct challenges to project leaders' authority and more „strategic forking“, namely more deviations from the main development path motivated by ambition for leadership.“ (Rossi 2004, S. 3f).

Was Maria Alessandra Rossi hier in teilweise sehr technischen Termini beschreibt, soll kurz erläutert werden:

Der erste Einwand betrifft die Teilnahme von hochqualifizierten Programmierern an Freier Software. Weshalb sollten diese, wenn Reputationszugewinne ihr primäres Ziel sind, ihre Arbeit frei zur Verfügung stellen, anstatt sich mit ihrer Idee auf den konventionellen Markt der proprietären Software zu begeben? Wiederum ist es die isolierte Betrachtung der Einzel-Motivationen, die Lerner und Tirole zu diesem Schluß veranlaßt hat – und Maria Alessandra Rossi zur Widerlegung desselben brachte[101].

Ein Aspekt, der im Rahmen der vorliegenden Literatur gänzlich vernachlässigt wird, ist die Frage nach der unterschiedlichen Wertigkeit von Reputation in Abhängigkeit der primären „Zielgruppe“, an welche die Reputationssignale gerichtet sind. Demnach ließe sich das Bedürfnis nach Anerkennung für den von Maria Alessandra Rossi angesprochenen „top-notch programmer“ [dt: erstklassig] lediglich dadurch befriedigen, indem er die Anerkennung aus einem ihm in Können und Wertschätzung mindestens gleichgestellten Lager bekommt. Anders formuliert: Reputation ist nur von Wert, wenn man sie von einer Person, einer Gruppe, empfängt, der man selber Anerkennung zollt. Unter dieser Perspektive entstehen völlig neue Konstellationen, welche noch mehr von der individuellen Verwobenheit verschiedener Motivationen abhängig sind. In erster Linie fallen die hauptsächlich ideologisch motivierten Freie Software-Entwickler in diese Kategorie. Diese können zwar Anerkennung für ihre technischen Fertigkeiten zum Beispiel auch aus dem proprietären „Lager“ entgegennehmen, aber ihr Bedürfnis nach Anerkennung für ihr Engagement in Freier Software kann niemals durch dieses „proprietäre Lager“ gestillt werden.

Der zweite Einwand besagt, daß, sollte Reputation als einzige Motivation angenommen werden, wesentlich häufiger eine Konkurrenz um „Führungspositionen“, also Projektleitung und →Maintainership, beobachtbar sein. Man könnte nun einwenden, daß ein potentieller →Kontributor in diesem Falle eher ein neues Projekt, welches „auf seinem Namen läuft“, initiiert. Allerdings gilt auch hier die Erkenntnis von Lerner und Tirole, daß nur im Falls eines erwarteten „net-benefits“ (Lerner/Tirole 2000, S. 16) ein entsprechendes Verhalten gewählt wird. Es muß sich demnach für den →Kontributor lohnen, eigene Ressourcen zu investieren. Ist dieser auf Anerkennung und Ansehen aus, kann es sehr viel lohnender sein, an einem erfolgreichen und eingeführten Projekt teilzunehmen und sich die Reputation „zu teilen“, als ein neues Projekt ins Leben zu rufen und dieses an einem sehr anspruchsvollen Markt zu etablieren.

Der dritte Einwand betrifft zudem die „technische“ Seite von abweichendem Verhalten. Maria Alessandra Rossi spricht von „strategic forking“ (Rossi 2004, S. 3), um es von technischem Forking abzugrenzen. Technisches Forking beschreibt den Vorgang der Abspaltung von Entwicklungszweigen eines Software-Projekts aufgrund technischer Überlegungen, beispielsweise grundlegenden Design-Entscheidungen. Technisches Forking ist – nach Ausschöpfung aller Mittel zur gütlichen Beilegung der Kontroverse – das letzte und auch adäquate Mittel zur Fortführung eines Projekts, mit dem Unterschied, daß nun das „originale“ und das „abgeleitete“ Projekt sich in technisch verschiedene Richtungen entwickeln. Diese Art des Forking wird als letztes Mittel akzeptiert, dient es doch dem Weiterbestehen eines Projekts auf einem technisch hohen Niveau. Im Laufe der Zeit entscheiden letztlich die Nutzer, welches der Projekte in der öffentlichen Diskussion besteht und welches in der Versenkung verschwindet[102]. Ein aktuelles, und auch innerhalb der Freien Software-Community für Aufsehen erregendes Forking war die Aufspaltung des X-Projekts[103] aufgrund lizenzrechtlicher Unstimmigkeiten. Das Team der X-En-

twickler entschied sich, die im Februar 2004 veröffentlichte Version 4.4 des X-Fenster-Systems mit einer „→Advertising-“ Klausel[104] zu versehen. Durch diese neue Klausel wurde die „XFree86-License“[105] als nicht mehr kompatibel zur →GPL erachtet und konnte somit nicht mehr im Verbund anderer, der →GPL unterstellter Software vertrieben werden. Die teilweise sehr heftig geführte Debatte innerhalb der Freien Software-Community – bei dem →XFree86-Projekt handelt es sich um einen zentralen Bestandteil eines graphischen, GNU/Linux-basierenden →Betriebssystems – führte letztlich dazu, daß sich ein Teil der Entwickler abspalteten und einen neuen Entwicklungszweig eröffneten[106]. Inzwischen hat das „neue“ Projekt bei Entwicklern wie Anwendern – hier insbesondere den großen →Distribution wie SUSE oder RedHat[107] – die Oberhand gewonnen, steht es doch für eine „freie“ und technisch überlegene Lösung.

Im Gegensatz zu diesem „technischen Forking“ existiert – theoretisch – auch die „strategische“ Variante, die von Rossi angesprochen wurde. Hier orientiert sich die Motivation, ein Projekt aufzuspalten nicht an technischen Details, sondern dient – um es umgangssprachlich auszudrücken – dem „entern“ oder „hijacken“[108] eines Projektes, um dessen Führung an sich zu reißen und in der Folge die Früchte genießen zu können, welche von Dritten gepflanzt wurden. Wie Rossi zurecht anmerkt, ist diese Form des Forking allerdings in der Praxis nicht anzutreffen (Vgl.: Rossi 2004, S. 4), was die oftmals postulierte Vorherrschaft des Reputations-Motivs weiter einschränkt. Um auf die Analogie Merton's zurückzukommen, wäre dieses Hijacking eine schwere Verletzung des Prinzips wissenschaftlicher Wissensproduktion.

Auch das teilweise zu beobachtende Insistieren auf der motivierenden Wirkung von „extrinsischen Aspekten“, beispielsweise monetärer Entlohnung, ist zumindest zu hinterfragen. Empirisch läßt sich nachweisen, daß der intrinsische „Wert“ einer Tätigkeit darunter leiden kann, daß zusätzliche extrinsische Anreize geschaffen werden. Alfie Kohn faßt die diesbezüglichen Erkenntnisse folgendermaßen zusammen:

„Researchers offer several explanations for their surprising findings about rewards and performance.

First, rewards encourage people to focus narrowly on a task, to do it as quickly as possible and to take few risks. ``If they feel that 'this is something I have to get through to get the prize,' they're going to be less creative,'' Amabile said.

Second, people come to see themselves as being controlled by the reward. They feel less autonomous, and this may interfere with performance. ``To the extent one's experience of being self-determined is limited,'' said Richard Ryan, associate psychology professor at the University of Rochester, ``one's creativity will be reduced as well.''

Finally, extrinsic rewards can erode intrinsic interest. People who see themselves as working for money, approval or competitive success find their tasks less pleasurable, and therefore do not do them as well.“ (Kohn 1987, S. 2)

Kohn bezieht sich in seiner Zusammenfassung auf Arbeiten von Richard Ryan und Edward Deci sowie Teresa Amabile – zwei Psychologen und eine Professorin an der Harvard Business School. Diese Autoren finden wir auch in weiteren wichtigen Artikeln um die Thematik „Freie Software“, meist bezüglich des Verhältnisses von extrinsischer und intrinsischer Motivation und der Auswirkung dieser beiden Motivationen auf das „well-being“ des Individuums. Insbesondere Ryan und Deci gehen auf den verstärkenden Zusammenhang zwischen einer intrinsisch anregenden

Tätigkeit, Autonomie und der Arbeit in der Gruppe ein (Vgl.: Ryan/Deci 2000, S. 9). Teresa Amabile betont die negativen Auswirkungen von „klassischen Arbeitsumgebungen“, die durch feste Hierarchien und „time-pressure“ für die Kreativität (Vgl.: Amabile 2002, S. 5) entstehen. In einem organisations-theoretischen Artikel geht sie näher auf die Faktoren ein, die eine der Kreativität förderliche Arbeitsumgebung ausmachen. Zu der Relation von extrinsischen Motivatoren und Kreativität – oder Produktivität – faßt sie zusammen, daß „[d]eciding how much time and money to give to a team or project is a judgment call that can either support or kill creativity.“ (Amabile 1998, S. 7)

Auch wenn diese drei Autoren kein Freies Software-Entwicklungsmodell im Auge hatten, sondern eher eine Art „praktische Grundlagenforschung“ zu arbeitsökonomischen Themen leisteten, so kann man große Bereiche ihrer Erkenntnisse und Vorschläge auf das verteilte Freie Software-Entwicklungsmodell übertragen – mehr noch: wir finden vieles davon schon verwirklicht in der dezentralen Struktur, der weitgehenden Autonomie und Verantwortung in und für die eigene Arbeit. Auch sind die verstärkenden „Kontextfaktoren“[109], insbesondere der intensive, nicht- oder flach-hierarchische Community-Gedanke, in einer fast idealen Ausprägung vorhanden.

Ein weiterer Einwand betrifft die speziell auf „delayed-payoffs“-ausgerichtete Suche nach Reputation. Die kulturelle Nähe zu einem akademischen Entwicklungs- und Austauschsystems zeigt teilweise die selben Konsequenzen in Bezug auf die Wahl der Verwertungsalternativen für das eigene Schaffen. In seiner sehr lesenswerten Einleitung des inzwischen zum Klassiker avancierten Buches „Open Sources: Voices from the Open Source Revolution“, die sich über 17 Seiten erstreckt und ein umfassendes und plastisches Bild von Entstehung und Philosophie von Open Source- und Freier Software zeichnet, faßt Chris DiBona seine Gedankten zum „Reputation game“ (DiBona 1999, S. 17) so zusammen, daß „[...] history has shown that scientific success outlives financial success.“ (DiBona 1999, S. 17). Auch wenn DiBona hier – in Anlehnung an Eric Raymond's „Homesteading the Noosphere“ (Vgl.: Raymond 2001b, S. 92f) – meiner Meinung nach etwas weit geht und seine Aussage damit belegen will, daß er die Reputation von historischen Wirtschaftsgrößen und berühmten Wissenschaftlern vergleicht. Auch wenn die Reputation, die man durch eine akademische Tätigkeit erwirbt, wohl eher von Dauer ist als eine Reputation, die sich allein auf wirtschaftlichen Erfolg – DiBona nennt im diesem Zusammenhang Namen wie Rockefeller oder Carnegie – und insbesondere gemessen an monetären Einheiten, beruft: in diesen Dimensionen spielt sich Freie Software-Entwicklung in der Regel nicht ab. Daher ist es in Bezug auf Freie Software-Motivation nicht die Aussicht, ein „[...] Einstein, Edison ... Pauling.“ (DiBona 1999, S. 17) zu werden, sondern vielmehr innerhalb „seiner“ Community ein „Einstein im Kleinen“ zu werden. Die Chancen, durch das eigene Schaffen weit über den Horizont der eigenen Community hinauszustrahlen sind äußerst gering – wie wir auch anhand der relativ geringen Anzahl an „öffentlich bekannten“ Freien Software-Entwicklern ablesen können. An diesem Punkt wird auch klar, daß die Teilnahme an einem Freien Software-Projekt bei der Befriedigung von Anerkennungsbedürfnissen wesentlich vorteilhafter ist als die Arbeit in und an proprietärer Software: in letzterem Fall fällt eine mögliche positive Reputation lediglich auf das Unternehmen, welches die Software publiziert zurück, der einzelne Entwickler bleibt in den meisten Fällen anonym.

So gilt auch hier, daß eine realistische Kosten-Nutzen-Bilanz meist negativ bezüglich des Strebens nach „globaler“ Anerkennung ausfällt – und fast immer im Falle einer im Voraus zu fällenden Entscheidung über ein bestimmtes Engagement. Um auf eine der wenigen Persönlichkeiten zurückzugreifen, die es tatsächlich schaffte, sich auch außerhalb einer sehr speziellen

Community einen Namen zu machen: Auch Linus Torvalds hatte 1991 nie eine Verbreitung seines →Betriebssystems – oder besser: seines rudimentären →Kernels, der es 1991 war – über die Grenzen der →Hacker-Kultur hinaus gedacht. In seiner Ankündigung über die →Mailing-Liste ist zu erfahren, daß er „doing a (free) operating system (just a hobby, won't be big and professional like gnu)“ (Torvalds 1991, S. 1; vgl.: Kapitel 2.1.5). Auch über den Wirkungskreis seines Schaffens – wie eben erwähnt – ist er sich im Klaren und behauptet von seinem Code, daß es ein „program for hackers by a hacker“ (ebd.) ist.

Dieser Einwand betrifft somit nicht die Motivation „Reputation“ an sich, sondern lediglich die Unterstellung, daß durch Freie Software-Entwicklung eine Reputation auch in angrenzenden Bereichen angestrebt wird. Die Reputation innerhalb der entsprechenden Community, die Reputation von und durch die primäre „peer-group“ allein ist als Erklärung für ein Engagement in Freier Software „stark“ genug (Vgl.: Kapitel 5.3).

Sanktionierungsmechanismen

Reputation ist ein knappes Gut und muß geschützt werden. Ebenso wie es auf der Seite von proprietärer Software Mechanismen zum Schutz von geistigem Eigentum gibt – hier technisch betrachtet durch das Vertreiben von →Binärversionen[110] – findet man auch im Bereich der Freien Software ein ausgeklügeltes System von Reputations-Zuweisung und Reputations-Sicherung. Behandelt wird nicht die übergeordnete institutionelle Sicherung durch beispielsweise das Urheberrecht, sondern die Normen und Regeln, welche sich innerhalb der Freien Software-Community herausgebildet haben. Formal funktioniert die Zuschreibung von Urheberschaft über die CREDIT-Datei, welche im →Quellcode jedes Software-Projekts zu finden ist, und in welche sich die Entwickler einschreiben. Die Entwicklung eines Projektes wird in der CHANGELOG-Datei protokolliert, es wird der „→Contributor“ [dt.: Beitragende] und der Beitrag genannt. Ein weiterer essentieller Bestandteil ist die Lizenz. Im Unterschied zu beispielsweise einer restriktiven EULA [dt.: End User License Agreement], deren Hauptzweck es ist, bestimmte Nutzung- und Verbreitungsformen zu unterbinden, also auf das Copyright zu bestehen, drehen Freie Software-Lizenzen diesen Umstand um: Nehmen wir als Beispiel die am weitesten verbreitetste[111] Freie Software-Lizenz, die →GPL, so insistiert sie auf dem „→Copyleft“, der unbedingten Beibehaltung der Freiheiten, und droht mit dem Entzug der Lizenz bei Nicht-Einhaltung dieser Freiheiten. Die Offenheit und somit allgemeinen „Bekanntheit“ des Codes in der Community sowie die Transparenz der Interaktionen der Gruppenmitglieder ermöglichen ein sehr schnelles und sehr genaues Matching von „Schädigung“ ihrer Werte und den entsprechenden devianten Mitgliedern.

Für Verstöße gegen die Lizenz bestehen Sanktionen nach bürgerlichem Recht – wie aber lassen sich Verstöße innerhalb der Community sanktionieren, wenn kein „absolutes“ ausführendes Organ vorhanden ist?

Um diese Frage zu beantworten, gilt es zuerst zu prüfen, was überhaupt durch Sanktionen „genommen“ werden kann und worin dieses „etwas“ seinen Ursprung hat: die Community kann nur entziehen, was sie zuvor gegeben hat. In diesem Falle ist es jene Reputation, welche im Freien Software-Umfeld von Community-Mitgliedern erworben wurde, und durch diese auch wieder entzogen werden kann. Dem Autor liegt jedoch kein derartiger Fall vor. Der Entzug läuft über die selben Kanäle wie die Zuschreibung von Reputation: die offenen Kommunikationsnetze der Freien Software-Entwickler wie →Mailing-Listen, →IRC, →ICQ oder →Foren, um nur einige zu nennen. Letztlich stehen zwei Möglichkeiten zur Sanktionierung bereit: das öffentliche Anprangern und das sogenannte „Flaming“, der Kommunikation in einem offensiven, absichtlich

abwertenden Ton. Beide Möglichkeiten sind darauf bedacht, den Norm-Brecher aus der Community auszuschließen und ihm somit die Grundlage seiner Suche nach Bestätigung und Ansehen zu entziehen. Osterloh spricht diesbezüglich, zusätzlich zum „Flaming“ von „kill-filing“ [dt.: Bekanntmachung, daß man keine Email von einer bestimmten Person erhalten will] und „shunning“ [dt.: vorsätzliches Nichtbeantworten von Emails] (Osterloh et al. 2002, S. 16). Letztlich laufen „kill-filing“ wie auch „shunning“ auf Kommunikationsverweigerung hinaus.

Da jede der angesprochenen Sanktionierungsmaßnahmen lediglich ein Gefühl von Scham induzieren kann, können damit ausschließlich intrinsisch motivierte Community-Mitglieder geahndet werden, die ihre „Freie Software-Existenz“ auf den Werten der Community aufgebaut haben. Wie Osterloh treffend bemerkt: „Purely extrinsically motivated egoists would not feel any shame.“ (Osterloh et al. 2002, S. 16, in Anlehnung an: Orr 2001, S. 57)

In den Diskussionen, welche im Vorfeld und während dieser Arbeit stattfanden, wurde oft die Frage angerissen, warum ein derartiges Projekt nicht ständig „ausgebeutet“ wird, da die Institutionen für eine Ahndung von Norm-Brechern fehle. Wie wir in den vorigen Abschnitten festgestellt haben, stehen sehr wohl Mechanismen zum Schutze dieser „freien“ Arbeit bereit. Auf der einen Seite das bürgerliche Recht, welches Verstöße rechtskräftig ahnden kann, sowie die „soziale Sanktionierung“, die einen Großteil der Community-internen Verstöße abfangen kann. Für den ersten Punkt liefert uns die jüngste Vergangenheit Beispiele für eine erfolgreiche Sanktionierung externer Devianz: Der kommerzielle Netzwerkausrüster Sitecom[112] nutzte in einem seiner Produkte Teile des Freien Software-Projekts netfilter[113], ohne den Forderungen der unter der GPL-lizenzierten Software nachzukommen. Daraufhin wurde Sitecom von Harald Welte – Vertreter des netfilter-Projektes und Interviewpartner – vor dem Amtsgericht München auf Unterlassung aufgrund der Bestimmungen der GPL verklagt. Die Entscheidung des Gerichts vom 14.04.2004[114] ist die weltweit erste rechtswirksame Überprüfung der GPL vor einem Gericht. Dem „Sitecom-Fall“ folgten weitere, ebenfalls erfolgreiche Klagen gegen Verstöße gegen die GPL. Harald Welte erwähnt auch im Interview diesen im Grunde erfreulichen Umstand – jedoch betont er die persönliche Abneigung, daß man sich überhaupt mit derartigen Problemen beschäftigen muß. Er spricht den Verlust an Zeit und Energie durch derartige Rechtsstreitigkeiten an, und wie gut diese Energie in einer sinnvollen Tätigkeit wie der Programmierung von Freier Software aufgehoben wäre (Vgl.: Welte 2004, S. 10). Eine ähnlich pragmatische Einstellung ist auch bei den anderen Interviewpartner vorherrschend.

Es liegen wie erwähnt keine empirischen Befunde zu ernsthaften Verstößen und Sanktionierungsmaßnahmen innerhalb der Community vor. Bezüglich der Community-Umwelt-Beziehung stellen die netfilter-Gerichtsverfahren (s.o) und insbesondere das seit Anfang 2001 immer noch schwebende Verfahren von SCO[115] über angeblich gesthohlenen UNIX-Code und dessen Einbringung in →Linux eine eher unrühmliche Ausnahme dar.[116] Interessant an dem SCO-Fall ist die heftige Reaktion der Freien- und Open Source-Gemeinde. Obwohl die Klage gegen IBM[117] sowie den Netzwerkspezialisten Novell[118] gerichtet ist, reagierte die Freie Software-Community, da man sich der positiven Konnotation der eigenen Arbeit und dem Stolz auf das bisher Geleistete beraubt sah. Zur sachlichen Auseinandersetzung, Rechtfertigung und zu Dokumentationszwecken wurde eine eigne Webseite eingerichtet[119], die aus der Perspektive der Freien Software-Entwickler und Befürworter die Integrität ihrer Community wieder herstellen und die die ihres Erachtens nach falschen Vorwürfe seitens SCO zurückweisen soll.

Auf der anderen Seite ist die – korrekte – Weitergabe von Reputation streng geregelt. Es existieren allgemein anerkannte Regeln für die Adoption eines verwaisten Projektes, was, wenn

die Adoption gelingt, auch eine Übernahme von „fremder“ Reputation bedeutet (Vgl.: Raymond 2001b, S. 73ff sowie Osterloh et al. 2002, S. 15). Raymond greift zur Erklärung des Eigentumbegriffs in Bezug zu Freier Software auf die Locke'schen Auffassung von Grundbesitz und deren Übertragungsmöglichkeiten zurück. Hintergrund des Locke'schen Gedankens ist der Überfluß an Land zur Zeit der frühen Einwanderung in den USA, und die drei Möglichkeiten, die den Pionieren gegeben waren, um an Land zu kommen: die einfachste Variante ist das „Homesteading“, das In-Besitz-nehmen von bisher besitzlosem Land, die zweite Möglichkeit ist der „Transfer of Title“, die öffentliche Übertragung der Eigentümerschaft, oft geregelt durch Erbfolge oder Kaufverträge. Die letzte Möglichkeit besteht in der Übernahme von „verwaistem“ Land, allerdings nur nach sorgfältiger Prüfung. Diese dritte Möglichkeit zieht – wie das „Homesteading“ - die Besitzanmeldung und weitere Kultivierung des Landes nach sich. Analog stellen sich diese Prozesse in Freier Software dar[120].

Konflikte im Abhängigkeits-freien Raum

Nach den bisherigen Beschreibungen verschiedener Aspekte von Freier Software wie Entwicklung, Zusammenarbeit oder Identifikation ist eventuell der Eindruck entstanden, daß es sich um etwas wie einen paradiesischen Zustand im rechtsleeren Raum handelt – dem ist nicht so. Auch das Phänomen Freie Software funktioniert nur, da es bestimmte Regeln und Normen gibt, an die sich die Beteiligten – ob extern als reine Nutzer oder intern als Entwickler oder →Maintainer – halten müssen. Zwar existiert kein verbindliches Regelwerk, wie es in anderen gesellschaftlichen Bereichen schriftlich niedergelegt und rechtswirksam durchsetzbar ist, doch haben sich innerhalb dieser →Hacker-Kultur ebenfalls allgemein anerkannte Regeln und Normen durchgesetzt (Tuomi 2004, S. 1ff). Dies ist durchaus ein Umstand, der innerhalb der Community den Partizipierenden bewußt ist, oder, wie es Rossi ausdrückt: „Even casual observation of the working of F/OSS [Free/Open Source Software; Anm. d. Verf.] projects suggests the importance of implicit or explicit norms that define the 'hacker identity'“ (Rossi 2004, S. 7).

Das Paradebeispiel ist das Verbot, sich mit der Arbeit Dritter zu schmücken. Im Software-Bereich schreiben sich die offiziellen →Kontributoren eines Projekts in der sogenannten „Credits“-Datei ein, welche den Quellen der Software beigelegt wird. Übertragen in den alltäglichen Sprachgebrauch kann man von einer Art Besitzurkunde oder der Signatur eines Künstlers sprechen, welche im Laufe der Zeit auch die Entwicklungen des Programms und aktiven Entwickler eines Projektes widerspiegelt. Mutwillig falsche Veränderungen an dieser Datei, beispielsweise die Streichung eines Eintrags, gilt als eine der „Todsünden“. Nicht Norm-konformes Verhalten – beispielsweise eine solche Streichung – gilt es aber auch in der Freien Software-Welt zu ahnden. Die Möglichkeiten dazu bietet nun nicht der „rechtskräftige“ Einflußbereich der Legislative aufgrund des Urheberrechts, vielmehr begegnet die Community dem Norm-Brecher mit ihren eigenen Sanktionen. Sanktionen, die auf der Verweigerung eben jener Vergütung beruhen, von welcher auch die „Position“ des Beteiligten ausgeht: der Reputation[121]. Durch den systematischen Entzug von Reputations-generierendem Verhalten der Community gegenüber dem Norm-Brecher hat dieser nur noch sehr eingeschränkte Möglichkeiten innerhalb – dieser – Community als Mitglied wieder Fuß zu fassen. Durch die offenen Kommunikationskanäle einer Freien Software-Community wird der Verstoß sowie die Reaktion der Community auch für Dritte als ein transparenter Prozeß vollzogen, was dem Norm-Brecher den Einstieg bei weiteren Freien Software-Projekten erschweren sollte.

Das Streben nach Anerkennung ist eng mit dem Gedanken der Konkurrenz verknüpft, da die Zuschreibung von Reputation – in der Regel[122] – aufgrund der Qualität des verfaßten

→Quellcodes beruht und dieser ausschließlich in Abgrenzung zu den Programmier-Leistungn Dritter beruht. In diesem Sinne ist Konkurrenz auch in Freier Software denkbar, beispielsweise bei der Besetzung von wichtigen, öffentlichkeitswirksamen Positionen; innerhalb von Projekten um die Anerkennung des jeweiligen Projekt-Leiters oder Koordinators, welcher für die Aufnahme oder Ablehnung der eigenen Arbeit in den Code-Stamm des Projektes verantwortlich ist; aber auch zwischen einzelnen Projekten, welche die selbe Nische belegen. Damit sind drei Konkurrenz-Ebenen aufgespannt: Eine vertikal und eine horizontal orientierte Konkurrenz sowie eine Inter-Projekt-Konkurrenz. Diese Punkte werden in der vorliegenden Literatur nicht adressiert, obwohl wiederum die auch für Dritte transparente Kommunikation in und zwischen einzelnen Projekten eine externe Aufdeckung und Analyse ermöglichen würde.

Über die Gründe kann an dieser Stelle nur spekuliert werden. Insbesondere zwei Aspekte sind denkbar:

Erstens kann der Konkurrenz-Begriff innerhalb Freier Software und in der Umwelt von Freier Software, also bei Nicht-Mitgliedern, unterschiedlich wahrgenommen werden. In seiner umgangssprachlichen Deutung – welche lediglich die lexikalisierte Bedeutung als „Rivalität“ und „Wettbewerb“ (Duden 1990, S. 421) überdehnt – werden oft Bilder verwendet, die auf eine persönliche Konkurrenz, eine Rivalität zwischen Individuen anspielen. Dabei ist beispielsweise von Ellenbogen-Mentalität die Rede. Durch die starke Verbindung zwischen der Person und dem Gedanken der Rivalität wird ein hohes soziales Konfliktpotential generiert. In der Freien Software wird Konkurrenz anders wahrgenommen: Hier ist diese Verbindung aufgetrennt, Konkurrenz wird nicht als Konkurrenz zwischen Individuen, sondern als Konkurrenz zwischen Ideen[123] betrachtet. Diese „technische“ Perspektive von Konkurrenz wirkt sich direkt auf das Verhalten der involvierten Individuen aus, deren Umgang weitgehend frei von persönlicher Anfeindung bleiben kann. Das Feld, auf welchem Konkurrenz ausgetragen wird, verlagert sich von der interpersonalen Ebene auf die „technische“ Ebene. Der angesprochene Umgang der Individuen kann damit von dem Konflikt-Potential einer Ellenbogen-Mentalität freigehalten werden und ist demnach für externe Betrachter, welche das erstgenannte Verständnis von Konkurrenz verinnerlicht haben, nicht oder nur schwer sichtbar.

Zweitens liegt diese Nicht-Existenz – bzw. die Nicht-Sichtbarkeit – von Konkurrenz in der Freien Software-Sphäre eventuell daran, daß auf einen relativ homogen, über die Mitglieder verteilten Satz von „Grundwerten“ zurückgegriffen werden kann. Sollte die Qualität des produzierten →Quellcodes das Bewertungsmerkmal sein, und sei weiter Konkurrenz über diesen Quellcode ausgetragen, so sollte selbst innerhalb der Freien Software-Sphäre keine persönliche Form der Konkurrenz entstehen. Aus der Perspektive des Freien Software-Entwicklers ließe sich formulieren, daß, sollten alle Entwickler den Primat der Qualität des Quellcodes vertreten, werden sie den Quellcode zur Austragung von Konkurrenz heranziehen. Damit wiederum steht der Quellcode im Mittelpunkt der Rivalität und nicht das Individuum. Je besser sich die Gruppe darauf verständigt hat, über welche Mechanismen Konkurrenz ausgetragen wird, und je homogener die Gruppe diesen Wertekanon teilt, desto besser funktioniert die Abstraktion der Rivalität von dem Individuum.

Altruismus

Geht man von der lexikalisierten Bedeutung des Begriffs „Altruismus“ aus, versteht man darunter eine Verhaltensdisposition, die durch Selbstlosigkeit[124] und Rücksichtnahme[125] gegenüber Dritten geprägt ist. Weiter ist zu unterstellen, daß altruistisches Verhalten immer

freiwillig und intentional ist sowie keine externalen Vergünstigungen oder Entlohnung erwartet. Die Intentionalität der altruistischen Handlung richtet sich demnach auf das Wohl eines oder mehrerer Dritter. Um die im folgenden Kapitel aufgeführten Bedenken gegen diesen Altruismus-Begriff zu verdeutlichen, sei auf das gegensätzliche Begriffspaar Altruist – Egoist verwiesen. Hier wird besonders deutlich, welchen Idealismus ein Individuum aufzubringen hat, um ein rein altruistisches Verhalten zu verfolgen. Für ein Engagement in Freier Software werden Erklärungen abgelehnt, die auf dieser Interpretation von altruistischem Verhalten aufbauen.

Das selbe Problem wird mit der Frage nach der Authentizität altruistischer Motive adressiert: Aus einer sozial-psychologischen Perspektive kann von Altruismus als einer möglichen Verhaltensdisposition ausgegangen werden, deren Wahl nicht a priori festgelegt ist[126], sondern in einem Auswahlprozeß zwischen alternativen Verhaltensdisposition verhandelt wird. Im Zuge dieser Verhandlung sind rein altruistische Verhaltensweisen unwahrscheinlich, da die persönlichen Zielsetzungen und externe Erwartungen an das Individuum zu vielfältig sind. Im Rahmen der kognitiven Dissonanztheorie würde rein altruistisches Verhalten Dissonanz auslösen, da nicht alle dem betreffenden Individuum bekannten „Meinungen oder Kenntnisse“ (Weiner 1976, S. 152) in Übereinstimmung oder Konsonanz gebracht werden können (siehe auch Kapitel 5.3.2).

Im folgenden Kapitel wird die in dieser Arbeit verwendete Interpretation des Altruismus-Begriffs dargelegt.

Herleitung von reziprokem Altruismus

Wir finden in verschiedenen wissenschaftlichen Disziplinen Erklärungsversuche zu altruistischem Verhalten. In der Biologie ist von Mutualismus, der Symbiose zum gegenseitigen Vorteil die Rede, in der Soziologie kennt man ähnliches aus der Beschreibung von Tausch- und Geschenkkulturen.

Um von „echtem“ altruistischem Verhalten zu sprechen, müßten die tatsächlichen Beweggründe der Beteiligten bekannt sein. Dies ist hier nicht der Fall, auch stellt sich generell die Frage nach der Validität einer solchen Erkenntnis: Altruismus ist ein sozial erwünschtes Verhalten, es ist somit sehr wahrscheinlich, daß entsprechendes Verhalten – auch wenn der Betroffene insgeheim mit bestimmten Gegenleistungen rechnet – weiterhin als altruistisch kommuniziert wird.

Die Psychologie hat den vielleicht „passendsten“ Begriff hierfür geprägt: den reziproken Altruismus (Trivers 1971 in: Rudolph 2003, S. 243). Seine Anwendung auf das Thema Freie Software ist mit weniger Einschränkungen bedacht wie beispielsweise die Raymond'sche Perspektive der Geschenkkultur (Raymond 2001b, S. 80). Raymond beschreibt das Geben und Nehmen in der Freien Software-Community als eine „gift-culture“ (ebd.) und vergleicht sie mit Transaktionsmustern, wie sie in Tauschkulturen und in bestimmten gesellschaftlichen Schichten, z.B. der extrem Vermögenden, anzutreffen sind. Ausgangspunkt ist der Gedanke des Überflusses, der Möglichkeiten der Distribution von Gütern und Ressourcen, die keiner Knappheit unterliegen. Die Unterstellung der fehlenden Knappheit ist allerdings problematisch: Raymond unterscheidet nicht zwischen dem Überfluß an grundsätzlich überlebenswichtigen Ressourcen wie Nahrung, und den Ressourcen, welche für die Produktion von Software bereitgestellt werden müssen. Mit seiner Argumentation kann er zwar eine Umwelt beschreiben, die der Produktion von Freier Software förderlich ist, jedoch noch nicht das Funktionieren des Freien Software-Entwicklungsmodells erklären. Ein anderer Einwand betrifft diesen Umstand des Überflusses: Freie

Software entstand nicht nur abgeschottet von der Außenwelt in einer Sphäre des „Luxus“, sondern verdankt ihre Existenz zum Großteil der frühen Unterstützung durch öffentliche Einrichtungen und privatwirtschaftlichen Unternehmungen.

Die Verbindung von Reziprozität und Altruismus ist in sofern interessant, weil sie zum Einen einen erweiterten zeitlichen Horizont der Interaktionen mit einbezieht, zum Anderen weil sie das starke ideologische Gewicht in Altruismus abschwächt. Der zweite Punkt betrifft die erwartete Balance zwischen eigener Einbringungs-Leistung und der Gegenleistung Dritter, was, sobald eine Gegenleistung in die Abwägung der eigenen Handlungsalternativen mit einbezogen wird, nicht mehr als „reiner“ Altruismus im Sinne von Selbstlosigkeit beschrieben werden kann. Der reziproke Altruismus ist, wie im Folgenden ausgeführt wird, besser als Erklärungsansatz für Verhalten in Freier Software geeignet.

Reziproker Altruismus im Freien Software-Entwicklungsmodell

Es kann ein Zusammenhang zwischen vermeintlich selbstlosem Verhalten und einer spezifischen Erwartungshaltung hergestellt werden: Altruismus oder altruistisches Verhalten bezieht sich daher in dieser Arbeit fast durchgängig auf die spezifische Form des reziproken Altruismus (s.o.). Der reziproke Altruismus versucht egoistische Prädispositionen und altruistisches Verhalten zusammenzubringen, indem er von einer verteilten sowie verzögerten Auszahlungsmatrix ausgeht. Darunter ist zu verstehen, daß derjenige, der sich vermeintlich altruistisch verhält, darauf hofft, daß man sich ihm gegenüber ebenfalls so verhalten wird. Diese Reziprozität ist aber nicht an das Individuum gebunden, für welches man beispielsweise Hilfe geleistet hat, sie definiert vielmehr eine Erwartungshaltung der Gemeinschaft gegenüber, in der sich in diesem Beispiel der Hilfeleistende und der Hilfesuchende aufhalten. Der Aspekt der verzögerten Auszahlung bezieht sich auf die zeitliche Dimension: Es muß kein direkter zeitlicher Zusammenhang zwischen Leistung und Gegenleistung bestehen, ebenso – wie schon erwähnt wurde – keine direkte Beziehung zwischen dem Hilfeleistenden und dem Hilfesuchenden bestehen muß. Je geschlossener die Gruppe ist, je niedriger die Fluktuationsrate der Mitglieder, desto eher wird sich ein effizientes System, welches auf einem reziproken altruistischem Verhalten beruht, installieren können. Hat die Gleichung der Auszahlungsmatrix zu viele Freiheitsgrade, wenn also die Chance für einen „Betrüger“ – ein Individuum, welches sich nicht für empfangene Hilfeleistung revanchiert – hoch ist, weiterhin auf Gruppenmitglieder zu treffen, welche noch nicht betrogen worden sind, dann wird sich parasitäres Verhalten in die Gruppe einschleichen.

Mit einer Freien Software-Community scheint eine genügend geschlossene und stabile Gruppe vorzuliegen, um dieses parasitäre Verhalten innerhalb der Gruppe zu minimieren. Auch experimentelle Befunde aus der Spieltheorie sprechen dafür, daß in einer solchen Gruppe der Gedanke der Kooperation die größten Chancen für eine Durchsetzung hat (siehe Kapitel 5.2.4).

Auch für altruistisches Verhalten in Freier Software muß die Vorbedingung erfüllt sein, daß der Nutzen der Teilnahme in der Freien Software-Community die Kosten – mindestens – deckt[127], denn „[E]ven among intrinsically motivated donators, martyrs and saints are in short supply.“ (Osterloh et al. 2002, S. 19). Speziell die Vorstellung des Überflusses in Software – nach Raymond der Raum aller denkbaren Gedanken (Raymond 2001b, S. 78) – sowie die Art der von den Entwicklern empfundenen Kosten, wirken als Push-Faktoren für Freie Software[128].

Die Boston Consulting Group[129] und die Open Source Technology Group[130] gingen unter anderem der Frage bezüglich der Kosten von Freier- und Open Source Software-Produktion nach. Es wurden Freie Software-Projekte unterschiedlicher Reife[131] untersucht, wobei 684 gültige Fälle

der Auswertung als Grundlage dienten. Die Befragten mußten dabei auf einer Scala von „Less important“ bis „Most important“ ihre Einschätzung zu vorgegebenen Kosten der Teilnahme an der Entwicklung von Freier Software abgeben. Die Arbeit an Freier Software auf Kosten des Schlafes wurde als „Most important“ an die Spitze der Kosten gesetzt, dicht gefolgt von dem Weniger an Zeit für soziale Kontakte (Lakhani/Wolf 2003, S. 18). Somit werden die Opportunitätskosten für Freie Software-Entwicklung als sehr gering empfunden. Die oftmals formulierte Motivation aus Gründen von „career concerns“[132] kann auch hier empirisch nicht bestätigt werden, rangiert doch auf der selben Skala die Wichtigkeit von „Professional/career advancement“ an vorletzter Stelle (Lakhani/Wolf 2003, S. 18).

Altruismus und die Verpflichtung

Altruismus ist als ein untergeordnetes Motiv für ein Engagement in Freier Software zu betrachten. Altruistisches Verhalten ist zum Einen keines der möglichen „primären“ Motivationen[133], wie beispielsweise die Reputation, zum Anderen ist Altruismus – besonders innerhalb der Freien Software-Community – stark mit dem Gedanken der Reziprozität verbunden und kann somit auch als Obligation, als empfundene Verpflichtung, betrachtet werden.

Versucht man, altruistisches Verhalten in Freier Software zu begründen, so stößt man auf die sozusagen soziale Seite des Altruismus, die Reziprozität. Da für diese Arbeit in Kapitel 5.2 „reiner“ Altruismus – verstanden als Selbstlosigkeit und als Gegenteil zu Egoismus – generell als Erklärung für das in Freier Software-Entwicklung beobachtete Verhalten abgelehnt wird, sei hier eine Herleitung von reziprokem Altruismus über eine neue Kategorie von intrinsischer Motivation – der Obligation – dargestellt:

Der Soziologe Siegwart Lindenberg führte in die Diskussion um intrinsische Motivation die Unterscheidung zwischen „enjoyment based“ und „obligation based“ (Lindenberg 2001, S. 318f) ein. Diese Unterscheidung verdeutlicht, daß auch intrinsische Motivation nicht immer sich selbst genügt, sondern manche Aspekte extern induziert werden. Somit entsteht neben der Selbstbestimmtheit eine weitere, durch das soziale Umfeld gesteuerte Motivation, sich im Sinne eines von der Community erwünschten Habitus zu verhalten. Diese Motivation bezeichnet Lindenberg mit „obligation based“, und spielt damit auf die Verpflichtung zur Reziprozität innerhalb der Community an. Diese Reziprozität wiederum läßt sich steuern und fördern. Dies geschieht im Bereich der Freien Software. Neben einem idealen Umfeld für altruistisches Verhalten im Sinne von geringen Kosten – verstanden als geringe Investition von materiellen Ressourcen und geringen Opportunitätskosten und daraus resultierend niedrigen Risiken durch die Teilnahme einen Verlust zu erleiden – sowie den technischen Möglichkeiten der zeit- und ortsunabhängigen, günstigen und dichten Kommunikation und der damit verbunden Transparenz, hat sich ein System institutionalisiert, welches ständig altruistisches Verhalten neu initiiert und fördert. Dabei wird altruistisches Verhalten, wenn es in einer Gruppe einmal verankert ist, zum „Selbstläufer“, da es unter den Mitgliedern ein wechselseitiges Gefühl der Verpflichtung schafft. Eine Verpflichtung, welche nicht direkt gegenüber dem „Spender“ gilt, sondern vielmehr dem gesichtslosen Ganzen der Community. Diese indirekten, nicht „einklagbaren“ Beziehungen haben den Vorteil, daß sie unaufdringlich sind und Enttäuschungen – außer in Extremfällen, beispielsweise der jahrelang geleisteten Hilfe und den immer fehlenden „Rückflüssen“ – vermeiden. Das System erhält sich selbst.

Eine weitere Rolle spielt die Institutionalisierung von altruistischem Verhalten durch die Gruppe, in der dieses Verhalten erwünscht wird. Ebenso wie beispielsweise im ehrenamtlichen Engagement Institutionen und Regeln existieren, welche das vorhandene Potential an altruistischem

Verhalten kanalisieren und organisieren[134], besitzt die Freie Software-Gemeinde Mechanismen zur Sicherung weiterer „Zuflüsse von altruistischem Verhalten“. Zum Einen wird die Attraktivität für das Beisteuern von altruistischem Verhalten erhöht, zum Anderen existieren Regelwerke zur Sicherung der „altruistischen Beiträge“ in dem Sinne, daß ein geleisteter Beitrag nicht später durch egoistisches Verhalten von anderen Gruppenmitgliedern oder Dritten, die nicht der Gruppe angehören, zweckentfremdet werden kann. Der erste Punkt wird zu einem Großteil durch die Erwartungshaltung der Gruppe für altruistisches Verhalten und somit durch einen gewissen sozialen Druck erreicht. Weiter besteht vor allem die Möglichkeit der Erlangung von Anerkennung (Vgl.: Kapitel 5.1) und der Befriedigung eines Zugehörigkeitsbedürfnisses (Vgl.: Kapitel 5.3.1.1). Der zweite Punkt wird im Wesentlichen durch die Lizenzierung[135] abgedeckt.

„Tit for Tat“

Die Kooperationsmuster einer Freien Software-Community lassen sich aus sozial-psychologischer Perspektive auch spieltheoretisch beschreiben. Innerhalb der Spieltheorie wurde das „[...] iterated Prisoner's Dilemma [zu dem] E. coli of social psychology [Hervorh. i. Orig.; Anm. d. Verf.].“ (Axelrod 1984, S. 28). Dieses „wiederholte Gefangenen-Dilemma“ geht auf den US-amerikanischen Politologen Robert Axelrod zurück, der es damit um eine zeitliche Dimension anreicherte. Das ursprüngliche Gefangenen-Dilemma, in welchem unterschiedliche Verhaltensmuster in einer virtuellen Umgebung gegeneinander „antraten“, endete mit dem Ergebnis, daß das einfachste Verhaltensmuster – hier: Strategie – gewann. Der verwendete Algorithmus wurde von seinem Schöpfer Anatol Rapoport „Tit for Tat“ getauft und beschreibt eine „nette“ [i. Orig.: „niceness“] (Axelrod 1984, S. 42), nicht nachtragende Strategie, die immer mit dem Verhalten reagiert, mit dem sie konfrontiert wird. Diese Strategie beginnt immer mit Kooperation und behält ein kooperatives Verhalten bei, bis die andere Partei „betrügt“. Tit for Tat reagiert auf Betrug mit einer einmaligen „Rückzahlung“, ebenfalls im Sinne von Betrug, um danach aber sofort wieder ein kooperatives Verhalten anzuwenden. Durch den schnellen Rückfall in kooperatives Verhalten wird Tit for Tat als nicht nachtragende Strategie bezeichnet. Ihr Erfolg innerhalb einer Population beruht in erster Linie darauf, daß sie wie erwähnt a priori nicht betrügt, und auf Betrug nur einmal mit Betrug, also nicht nachtragend reagiert. Tit for Tat ist umso erfolgreicher, je homogener die Population ist, in der Tit for Tat zur Anwendung kommt, und je größer der zeitliche Horizont der veranschlagten Interaktionsbeziehungen ist – zwei Punkte, die in der Entwicklung und dem Engagement in Freier Software gegeben sind.

Rudolph legt anhand „Tit for Tat“ dar, daß in einer Population, in der diese Strategie von eine Teilmenge praktiziert wird, diese niemals „aussterben“ (Rudolph 2003, S. 248) wird, da sie auf Dauer allen anderen denkbaren Strategien überlegen ist. Axelrod faßt zusammen, daß

„[I]f a nice strategy, such as TIT FOR TAT [Hervorh. i. Orig.; Anm. d. Verf.], does eventually become adopted by virtually everyone, the individuals using this nice strategy can afford to be generous in dealing with any others. In fact, a population of nice rules can also protect itself from clusters of individuals using any other strategy just as well as they can protect themselves against single individuals. [...] Cooperation can begin with small clusters. It can thrive with rules that are nice, provocable, and somewhat forgiving. And once established in a population, individuals using such discriminating strategies can protect themselves from invasion. The overall level of cooperation tends to go up and not down. In other words, the machinery for the evolution of cooperation contains a ratchet.“ (Axelrod 1984, S. 177)

Die Ausführungen Axelrod's finden wir in seinem Werk unter dem Kapitel „The Robustness of Reciprocity“ (ebd., S. 169-191), einer Einschätzung, der auch hier gefolgt wird. Doch bes-

chränkt sich der Erkenntnisgewinn nicht nur auf die Bestätigung des angenommenen reziproken Altruismus in Freier Software, vielmehr finden wir in Axelrod's zitierter Schlußfolgerung auch Verknüpfungen zu den Aspekten von Sanktionierung in Freier Software (Vgl.: Kapitel 5.1.4) und zum Erfolg des Community-basierten Entwicklungsmodells (Vgl.: Kapitel 5.3).

Wie lassen sich diese Überlegungen auf Freie Software-Entwicklung übertragen? Freie Software-Entwicklung wird manchmal als irrational bezeichnet (s.u.), hauptsächlich von Wirtschaftswissenschaftlern, für die dieses Organisationsprinzip scheinbar fundamentale Grundsätze untergräbt. Stellvertretend sei Prof. Dr. Schauenberg genannt, der in einem Gespräch zu dem Thema der vorliegenden Arbeit befand: „Die einzige Motivation, warum jemand so etwas [Entwicklung von Freier Software; Anm. d. Verf.] machen kann, ist die Irrationalität.“[136] Nun scheint aber die Motivation für Freie Software-Entwicklung nach der „Tit for Tat-“ Strategie unter dem rational choice-Paradigma einzuordnen zu sein, und ist damit nicht mehr „irrational“, sondern vielmehr „angebracht“.

Community

„It's the community, stupid.“[137]

Wenn über Freie Software-Entwicklung gesprochen wird, wird über „Community“ gesprochen[138]. Mit diesem Begriff soll die „zwischenmenschliche Art und Weise“ der Freien Software-Entwicklung zum Ausdruck gebracht werden; mit ihm soll ein neues „technisches“ Entwicklungsmodell beschrieben werden – und nicht zuletzt deutet der Begriff an, daß es Grenzen gibt, daß diese Community „Mitglieder“ sowie eine Art „Umwelt“ besitzt.

Betrachtet man den Begriff „Community“ genauer und versucht eine Übertragung ins Deutsche, so bieten sich am ehesten „Gemeinde“, „Gemeinschaft“ oder auch etwas formaler „Gemeinwesen“ an. Eine Definition von Community basiert immer auf einer spezifischen Gemeinsamkeit ihrer Mitglieder, wie beispielsweise dem Wohnsitz. Mit dieser Definition ist das deutsche Wort der Gemeinde als eine lokale Verwaltungseinheit gefaßt, jedoch nicht die Art von Gemeinde, welche im Zusammenhang mit Freier Software untersucht wird. Weiter als die geographische Definition führt die Betonung einer „ideellen“ Gemeinsamkeit. Auch hier wird man beim klassischen Gemeindebegriff fündig – wenn man sich eine Glaubens-Gemeinde oder die Gemeinde eines Künstlers vorstellt. Der erstgenannte Terminus betont den Glauben als das zusammenschweißende Element, der zweitgenannte ein gemeinsames Interesse. All dies für sich genommen macht die Gemeinde oder Community noch nicht aus, ein entscheidendes, bisher unerwähntes Element ist die Interaktion der Gemeindemitglieder untereinander. Bezüglich der Definition kann man einen starken Abfall der Interaktionshäufigkeit an den Grenzen der Gemeinde feststellen, also einer gegenüber der intra-gemeinschaftlichen Interaktion reduzierten extra-gemeinschaftlichen Interaktionshäufigkeit- und Intensität: dies ist bei Freier Software der Fall. Es liegen dazu keine empirischen Befunde vor, jedoch gibt es Hinweise für diese unterschiedlichen „Ideenwelten“, in denen die Community und ihre Umwelt leben. Die Existenz der Gruppe an sich bedingt schon – wie eben ausgeführt – die Existenz von Grenzen, zusätzlich läßt sich jedoch ein Konfliktpotential an diesen Grenzen feststellen. Dieses Konfliktpotential, hier eher verstanden als eine spezifische Einstellungs- und Meinungsdisposition gegenüber Nicht-Gruppenmitgliedern, manifestiert sich beispielsweise darin, daß es wertende Titel für bestimmte Mitglieder der Gruppe gibt[139], die Gruppe selber aber diese Wertung entweder nicht anerkennen oder sie – falls es sich um eine negative Wertung handelt – umdeuten. Dieser Um-

deutungsprozeß bestimmter Kommunikationsmuster zwischen der Gruppe und ihrer Umwelt kann mit den Überlegungen zur kognitiven Dissonanz (siehe Kapitel 5.3.2) gefaßt werden.

Auch eine etymologische Betrachtung des Begriffs paßt in die hier verwendete Interpretation von Freier Software. So werden die Wurzeln des Begriffs Community folgendermaßen beschrieben „[T]he word community comes from the Latin munus, which means the gift, and cum, which means together, among each other. Community literally means to give among each other. Community could be defined as a group of people who share gifts which they provide to all [Hervorh. i. Orig.; Anm. d. Verf.].“[140] Der Anspielung auf die Geschenkkultur folgt auch der „Open Source Evangelist“ Eric Raymond, der die Austauschbeziehungen der Open Source- und Freien Software-Gemeinde als Geschenkkultur und nicht als – ökonomische – Tauschkultur betrachtet. Raymond geht von der Prämisse der „Knappheit“ aus, welche für die Entstehung von Tauschkulturen verantwortlich ist. Weiter folgert Raymond, daß in der Freien Software-Kultur diesbezüglich andere Bedingungen herrschen und unter dem Paradigma des „abundance“ [dt.: Überfluß] (Raymond 2001b, S. 81) eine Tauschkultur ein „pointless game“ (ebd.) darstellen würde.

Geht man demnach für den Bereich der Freien Software von der Prämisse des Überflusses aus, so ist eine Tauschkultur kein adäquates Modell. Im Überfluß bleibt nur die Schenkung, wenn Güter weitergegeben werden sollen. Bekommt man in der Tauschkultur für ein Gut einen Preis, der die Qualität sowie die Nachfrage nach dem Gut widerspiegelt, so bekommt man in der Geschenkkultur Anerkennung und Ansehen für sein Geschenk, deren Höhe sich wiederum an der Qualität und der Nachfrage orientiert. Raymond stellt daher zu Recht fest, daß es in der Freien Software-Sphäre konsistente Muster gibt, über welche die jeweiligen Beiträge durch Reputation gewürdigt werden[141].

Bedürfnis nach sozialer Validierung

Wir haben in Kapitel 5.1 schon die Verknüpfung zwischen Reputation und der Gruppe angesprochen. An dieser Stelle wollen wir diese Verbindung aus der Perspektive des Bedürfnisses nach Autorität nach Heinrich Popitz neu beleuchten. Da für die Ausführungen Popitz' die Frage der „sozialen Validierung“ (Popitz 1987, S. 633) ein zentrales Thema darstellt, ist dieser Punkt unter dem Community-Aspekt eingeordnet, auch wenn eine Interpretation aus Sicht der Reputation ebenso möglich wäre.

Auch Butler et al. ordnen die Motivation der Schaffung und Erhaltung der Community den anderen Motivationen über. In ihrer Arbeit über die Bildung von Online-Communities gehen sie davon aus, daß „[...] expectation of these four types of benefits: informational, social, visibility, and altruistic, provide a range of motivations for why people do the work of online community building.“ (Butler et al. 2002, S. 11). Ihre Perspektive setzt weniger auf den „personal benefits“ (ebd.) auf, sondern vielmehr auf der Konstellation der freiwilligen Verbindung der Gruppenmitglieder untereinander, denen Butler et al. unterstellen, daß ihr Handeln entweder auf die Hilfe „ihrer“ Gruppe, oder aber auch eine größere Gruppe, welche „ihre“ Gruppe enthält, gerichtet ist. Somit verbindet sich hier der Community-Aspekt mit der in Kapitel 5.2 beschriebenen Motivation einer altruistischen Disposition.

Popitz versteht unter „sozialer Validierung“ die Existenz einer sozialen Bezugsgröße, die für den „Außenhalt“ (Popitz 1987, S. 633), für die „Bestätigung durch Andere“ (ebd.) verantwortlich ist. Popitz erweitert allerdings die Frage nach der sozialen Validierung um das Konstrukt der „dop-

pelten Anerkennung“ (ebd.), indem er das Zusammenspiel zwischen dem Respekt gegenüber der Autorität sowie dem Wunsche nach Anerkennung durch diese Autorität beschreibt.

Popitz und die Zugehörigkeit

„Werden wir uns nicht erst in den Blicken, die ein Anderer auf uns wirft, unserer selbst bewußt?“[142]

Popitz folgt in seiner Analyse über den Wandel der Autoritätsbedürfnisse einer linearen geschichtlichen Betrachtungsweise. Die Frage, die ihn dabei leitet, ist, in welcher Weise das Individuum nach sozialer Anerkennung sucht. Man findet Parallelen zwischen dem Community-Gedanke in der Freien Software als auch in Popitz' „ältester“ Form sozialer Anerkennung: dem Zugehörigkeitsgefühl.

Diese „Erfahrung der Zugehörigkeit“ (ebd., S. 638) stellt einen grundlegenden Bestandteil sozialer Erfahrung dar und gehört, wenn aus der Erfahrung „Gewißheit“ (ebd., S. 638) wird, wenn die immer wieder von Neuem gesuchte Erfahrung Bestätigung durch Dritte erfährt, zu „einer Grundform sozialer Selbstbestätigung“ (ebd., S. 638). Auch wenn die Zugehörigkeit zu einer Gruppe per se gegeben ist, so muß doch jeder Einzelne etwas leisten, um Anerkennung für seinen Platz in dieser Gruppe zu bekommen. Popitz nennt als Beispiele den Schutz der Gruppe, gegenseitige Hilfe innerhalb der Grenzen der Gruppe und die Teilnahme an kollektiven Arbeiten (Vgl.: ebd., S. 638).

Weiter behandelt er die Frage nach dem „woher“ der Anerkennung: „Von wem kann die „kritische“ [Hervorh. i. Orig.; Anm. d. Verf.] Anerkennung einer Zugehörigkeit erwartet werden?“ Popitz greift auf den historischen Entwurf der „institutionellen Autoritäten“ (ebd., S. 638) zurück, um diese Instanzen als Patriarchen oder Priester zu beschreiben. Weiter können auch Gruppen „als Ganzes“ Autorität ausüben, in diesem Falle generiert die Majorität der Gruppe eine spezifische Stimmung – beispielsweise der Anerkennung oder der Ablehnung – gegenüber einem dritten Gruppenmitglied und übt somit „als Ganzes“ Autorität auf diesen Dritten aus. Er stellt weiter sehr treffend fest, daß der Grad der Homogenität der Gruppe direkt mit der „Gruppenautorität“ (ebd., S. 638) korreliert und folgert daraus: „Jeder kann durch Zuwendung oder Abwendung […] in jedem anderen Zugehörigkeitssicherheit verstärkten oder Zugehörigkeitszweifel wecken.“ (ebd., S. 638). Popitz führt in diesem Zusammenhang die soziale Kontrolle ein, welche in durch Gruppenautorität geprägten Kollektiven einen zentralen Platz einnimmt und daß es keine Möglichkeit mehr gibt, sich zu „verstecken“.

Popitz, die Zugehörigkeit und Freie Software

Nachdem wir das grundlegende Konzept der Zugehörigkeit nach Popitz kennengelernt haben, möchte ich auf die angesprochenen Parallelen zwischen den Strukturen und Anerkennungs-Zuweisungen in Popitz' allgemeiner Ausführung und der speziellen Ausprägung dieses Gedankens in Freier Software eingehen.

Popitz' Ausführungen zur Gruppenautorität und der „Zugehörigkeit [als] eine Grundform sozialer Selbstbestätigung“ (ebd., S. 638) decken sich mit den Beobachtungen der – idealtypischen – Freien Software-Community. Folgen wir seinen Ausführungen der Reihe nach – wobei der initiale Moment für die Entscheidung der Teilnahme an einem Freien Software-Projekt nicht nach Popitz beantwortet werden kann.

Wir gehen davon aus, daß das grundlegende Verlangen nach sozialer Einbettung bei Freien Software-Entwicklern hauptsächlich durch andere soziale Kollektive wie beispielsweise das Umfeld der Erwerbsarbeit oder Freizeitgruppen gedeckt ist, demnach dieses Verlangen für diese Gruppe – die Freien Software-Entwickler – nicht verantwortlich ist, daß sie sich „als Mitglied dieser Horde, dieser Sippe, dieses Stammes (und weiter: dieses Staates, dieser Kirche)“ (ebd., S. 637/638) bewerben. Der initiale Moment kann viel eher mit den hier behandelten Motivationen wie Reputation, Spaß oder dem Mangel erklärt werden[143]. Interessant werden die „Autoritätsbedürfnisse“, wenn wir uns auf den Zustand des Mitglied-Seins, der tatsächlichen Mitgliedschaft konzentrieren. Hier können wir alle drei essentiellen Punkte Popitz' auf Freie Software-Communities übertragen:

Erstens erfüllt die Freie Software-Community die Bedingung, daß die soziale Gruppe bestimmte Anforderungen an ihre Mitglieder stellt – Anforderungen, die sich sowohl als Verhaltenskodexe wie auch als Leistungen manifestieren können. In einem Freien Software-Projekt und der umgebenden Community gibt es klare Regeln, wie sich jedes einzelne Mitglied zu verhalten hat, in welcher Weise der Umgang mit Außenstehenden gepflegt wird und welche Kommunikationskanäle zu benutzen sind. Dieser Punkt wurde in Kapitel 5.1.4 dargestellt. Allerdings ist die Forderung von Leistung[144] nicht etwa fakultativ, wie es bei Popitz erscheint. Sie ist in der Freien Software-Community obligatorisch, da lediglich über sie eine Zugehörigkeit ermittelt werden kann. Es stehen keine angeborenen Mitgliedschaften für eine Freie Software-Community bereit: „Talk is cheap. Show me the code.“ (Torvalds 2000, S. 1). In Verbindung mit Popitz' dritten Punkt (s.u.) – der starken sozialen Kontrolle in Gruppen mit Gruppenautorität – kann diese geforderte Leistung in Form von Beiträgen durch Programmierung, Dokumentation oder der Pflege des Außenauftritts des Projektes auch nicht zurückgehalten werden, da die sozialen Kontrollmechanismen diese „Leistungszurückhaltung“[145] sofort aufdecken und mit Sanktionierung reagieren würden. In der Realität stellen sich diese Vorgänge in der Regel in sehr abgeschwächter Form dar, aufgefangen durch das offene, kommunikative Klima eines Freien Software-Projekts. Auch – wie wir insbesondere in Kapitel 5.5 erfahren haben – besitzt die eigene Tätigkeit in einem Projekt meist eine derart hohe intrinsische Motivation, daß Leistungszurückhaltung nur zu beobachten ist, wenn das eigene Engagement in dem entsprechenden Projekt schon aus eigenem Antrieb „zurückgefahren“ wird und ein Ausstieg aus der aktiven Teilnahme ansteht. Damit haben wir die zwei nächsten Punkte in Popitz' Abhandlung schon angerissen.

Zweitens gilt es, die Autoritätsbeziehungen in einer Freien Software-Community auf die Möglichkeiten der „Autoritätszuschreibung“ zu untersuchen. Popitz geht von drei Möglichkeiten aus, durch welche auf Instanzen Autorität übertragen werden kann: der institutionellen Autorität, der „einzelne[n] Person“ (Popitz 1987, S. 638) sowie der Gruppe als Ganzes.

Einschränkend ist zu bemerken, daß die institutionelle Autorität hier in Bezug auf das Community-Modell der Freien Software-Entwicklung zwiespältig erscheint. Zum Einen existieren keine klassischen „institutionellen“ Autoritäten in Freier Software, zum Anderen kann aber bestimmten Persönlichkeiten und Projekten ein patriarchaler Charakter nicht abgesprochen werden, was unter Popitz' zweiten Punkt, der Repräsentation der Autorität durch bestimmte, einzelne Gruppenmitglieder fällt.

So wird Linus Torvalds oft als „Benevolent [dt.: wohlwollend] Dictator“ (Vgl.: Rivlin 2003, S. 1ff) bezeichnet. Eine Bezeichnung, die humorvoll den Führungsstil beschreibt und das Vertrauen der „Anhänger“ in ihren „Führer“ zum Ausdruck bringt. Dieser Titel hat inzwischen einen festen

Platz in der Freien Software- oder Open Source-Kultur gefunden, und eine Reihe weiterer „Projekt-Koordinatoren“ können sich mit diesem Titel schmücken. Interessant scheint mir das Auseinanderfallen des „Patriarchats“ – verstanden als starker, Personen-zentrierter Führungsstil mit sozialem, „sich-kümmernden“ Charakter – und der Autorität, von welcher ein Gruppenmitglied Anerkennung erwarten kann. Nach Popitz ist dies eine Instanz, sozusagen in Personalunion. In Freier Software kann durchaus eine Gruppenautorität für die Entscheidungen zur Gruppenzugehörigkeit „zuständig“ sein, die Koordination des Projekts aber von dem Patriarch, hier beispielsweise dem „Benevolent Dictator“ (ebd.) ausgehen.

Drittens beschreibt Popitz die „Hüter“ der Autorität. Dabei stellt er zwei interessante Aspekte dar: Zum Einen das Phänomen der Gruppenautorität[146], zum Anderen die Totalität der sozialen Kontrolle in einer homogenen Gruppe, wobei der zweite Punkt als Konsequenz des ersten verstanden wird. Popitz argumentiert, daß eine zunehmende Homogenität der Gruppe dazu führt, daß dadurch „jedes Mitglied als Hüter der Gruppenzugehörigkeit fungieren“ (Popitz 1987, S. 638) kann und somit sanktionierend und kontrollierend auf andere Gruppenmitglieder einwirken kann. Die Sanktionierung kann – wie wir schon in Kapitel 5.1.4 gesehen haben – durch „Anerkennungsentzug“ oder „Zugehörigkeitszweifel“ (ebd.) durchgesetzt werden. Übertragen auf die Freie Software-Community interpretieren wir „Anerkennungsentzug“ als Verweigerung weiterer Reputation und Achtung dem devianten Mitglied gegenüber. Der Begriff des Entzuges ist hier in doppelter Hinsicht zu betrachten: Einerseits kann die Gruppe keine Anerkennung entziehen, welche sich das deviante Mitglied über die Grenzen dieser Gruppe hinaus erworben hat, andererseits greift der Begriff „Entzug“ eher schlecht in Bezug auf zukünftige Maßnahmen, wie der angesprochenen Verweigerung von Reputation in weiteren Interaktionen des devianten Mitglieds mit der Gruppe.

Die zweite Maßnahme, bei Popitz „Zugehörigkeitszweifel“ (ebd., S. 638), drängt in erster Linie dieses Mitglied aus der Community heraus, und schneidet somit den Zugang zum gemeinsamen „Pool“ von Reputation ab.

Popitz behandelt fünf Formen von „sozialer Subjektivität“[147] (ebd., S. 637) als geschichtliche Sequenz von Autoritätsbeziehungen, betont aber, daß aus der heutigen Perspektive diese „fünf Typen sozialer Subjektivität kumulieren“ (ebd., S. 647) und zu einer „Pluralität von Autoritätsbedürfnissen“ führen (ebd.). Damit decken sich seine Analysen mit den Beobachtungen der Freien Software-Entwicklung: Man erkennt eine Evolution der Motivationen, bei der die anfänglichen Motivationen mit zunehmender Einbindung in eine Gruppe erodieren und neue Motivationen, neue Bedürfnisse entstehen.

Popitz fünfte und letzte Form der sozialen Subjektivität, die Anerkennung der sozialen Individualität, ist nicht mehr ohne weiteres mit den bisher festgestellten Parallelen zu vereinbaren. Gerade die von ihm ausgemachte Ursache für den fortgeschrittenen Drang zur Individualisierung, die andauernde „Konfrontation mit dem Andersartigen“ (ebd., S. 642) durch die städtische Agglomeration, kann für die Beschreibung einer Freien Software-Community nicht angenommen werden, eher das Gegenteil ist der Fall. Meines Erachtens ist dieses – bisherige – Auseinanderfallen der Erklärungsansätze in diesem Fall vor allem darin begründet, daß die Freie Software-Community sich noch vor ihrer Umwelt in vielen Belangen „schützen“ muß sowie innerhalb der Gruppe eine weitaus höhere Harmonie bezüglich der geteilten Wertevorstellungen und Wünsche herrscht. Mit anderen Worten: Die empfundene Sicherheit gegenüber der Umwelt der Freien Software-Community ist – noch – nicht so groß, als daß innerhalb der Community die Tendenzen zum „Auch-Dazugehören“ (ebd.) abgelöst werden von individualisierenden Tenden-

zen, welche das schützende Konstrukt der Gruppe zerstören würde. Solange das Individuum diese Sicherheit, diese Anerkennung in der Interaktion mit und in der Umwelt seiner Gruppe nicht empfindet, solange wird es nicht die Transformation zu Popitz' fünften Typus sozialer Subjektivität durchlaufen.

Stabilität einer Freien Software-Community

So wie Popitz eine zunehmende Tendenz der Individualisierung thematisiert (s.o.), welche eine Gruppenautorität – wenn auch nicht völlig verdrängt, aber doch einen immer größeren Stellenwert einnimmt, so bleibt die Frage nach der Stabilität des Konstruktes der Freien Software-Community als Gruppe. Daß individualisierende Tendenzen innerhalb der Freien Software-Community momentan keine Bedrohung dieses Entwurfs als Gruppe darstellen, wurde im vorigen Abschnitt erläutert (Vgl.: Kapitel 5.3.1.2). Es bestehen aber noch weitere „Bedrohungen“, weitere Fragen hinsichtlich der Stabilität der Gruppe[148] in Freier Software-Entwicklung.

Die erste Frage adressiert den Begriff der Stabilität in Gruppenprozessen. In der vorliegenden Arbeit soll eine Gruppe als stabil bezeichnet werden, wenn sie über einen längeren Zeitraum besteht und sie sich die ursprünglich geteilten Interessen, Ziele und die geteilten Wertevorstellungen und Ideale, auf welche sich das Wir-Gefühl stützt, erhalten kann. Weiter muß sie internen wie externen Bedrohungen und Belastungen standhalten können. Dieser Punkt ist lediglich im Fall einer tatsächlich beobachtbaren Bedrohung einer weiteren Analyse zugänglich. Für den Fall der Freien Software wurden derartige Bedrohungen nicht beobachtet.

Die zweite Frage adressiert den Grad an Stabilität, der bei Freien Software-Projekten tatsächlich zu beobachten ist. Betrachtet man den zeitlichen Aspekt, so können die meisten Freien Software-Projekte, die einer breiten Öffentlichkeit auch außerhalb der Software-Entwicklungs-Sphäre bekannt sind, auf eine über 10-jährige Geschichte zurückblicken. Beispielhaft sei das →GNU-Projekt, der Linux-Kernel, der Samba-Dateiserver und der Apache-Webserver[149] genannt. Die Stabilität der Werteorientierungen läßt sich dagegen nicht derart problemlos verifizieren. Da hierzu keine Literatur vorliegt, sei darauf hingewiesen, daß alle der genannten – und unzählige weitere – Projekte auch nach über 10 Jahren und einem exponentiellem Wachstum der Projekt-Mitglieder, wie auch des wirtschaftlichen Erfolgs, immer noch Freie Software sind und in ihrem jeweiligen Markt noch immer eine dominante Stellung einnehmen. Die Evolution der Lizenz kann als Indikator für die Werteorientierungen der jeweiligen Projekt-Mitarbeiter herangezogen werden. Wie in Kapitel 5.1.3 berichtet, ist auch ein prominenter Fall – hier: →XFree86 – bekannt, in welchem im Laufe der Zeit andere Schwerpunkte als die Freiheit dieser Software gesetzt wurden.

Nachdem man wie ausgeführt eine Freie Software-Community in der Regel als ein stabiles Konstrukt bezeichnen kann, adressiert die dritte Frage die Mechanismen oder Umstände, welche für Stabilität in einer Freien Software-Community sorgen. Neben dem in Kapitel 5.3.1 angesprochenen Bedürfnis nach dem Gefühl der Zugehörigkeit und dem im folgenden Kapitel beschriebenen Engagement für Freie Software aus ideologischen Gründen, welche über die Zeit als stabil betrachtet werden, können weitere Mechanismen zur Sicherung der Stabilität festgestellt werden.

Aus eigener Erfahrung und Beobachtungen von „Novizen“ in Freier Software möchte ich hier von einer „Spirale der sozialen Einbindung“ sprechen. Das Bild der Spirale steht für eine Art Automatismus, der dafür verantwortlich ist, daß Interessenten und Mitglieder der Freien Software-Sphäre immer stärker als Mitglied einer entsprechenden Community eingebundenen wer-

den. Insbesondere zwei Punkte stehen damit in unmittelbarem Zusammenhang: Einmal die Attraktivität des Modells an sich, welches – wenn es auf bestimmte Dispositionen, auf Potential das nach Bestätigung sucht, trifft – in seinen Möglichkeiten prädestiniert für die Befriedigung der Bedürfnisse nach Reputation, Zugehörigkeit und Spaß[150] erscheint; und weiter die Mechanismen, und Prozesse, welche das Verhalten und die Handlungen des „Novizen“ erfassen, sobald er den Eintritt in die Freie Software-Sphäre gewagt hat und zu einem Mitglied dieser avanciert ist. Denn sobald sich das Individuum als Mitglied der Gruppe begreift und vor allem auch von der Gruppe als Mitglied anerkannt wird, sobald also das Individuum in den Kanon des „Wir“ einstimmt, wird das Individuum die Gruppe als etwas begreifen, was ihn in seinem Weltbild und seinen Ausdrucksmöglichkeiten schützt. Das Individuum reagiert auf die Anerkennung als Mitglied mit einem Gefühl der Verantwortung und Solidarität mit den Werten und Zielen der Gruppe. Nun bietet eine Freie Software-Community auf der ideologischen[151] Ebene einen Satz von geteilten Werten, und auf technischer Ebene eine ideale →Plattform für persönlichen, kreativen Umgang mit Software. Hat das Mitglied dies verinnerlicht, wird es versuchen, sich seine Welt – die Community – zu erhalten. Weiter wird es die – notwendigen – Interaktionen der Community mit seiner Umwelt besonders kritisch beobachten, um den eigenen Entwurf gegebenenfalls vor negativem Einfluß zu schützen.

Die Gruppe für sich ist somit idealtypisch in einem konsonanten[152] Zustand. Erst durch Interaktion und Wahrnehmung einer Umwelt können Dissonanzen entstehen. Auf eine ausführliche Besprechung der Theorie der kognitiven Dissonanz wird hier verzichtet, da hauptsächlich die Konsequenzen aus diesem Erklärungsansatz hier von Bedeutung sind. Die Theorie behandelt die Unvereinbarkeit von Information und Erfahrung und der daraus resultierenden inneren Spannung. Dem Zustand der Dissonanz entspricht diese innere Spannung, dabei wird dieser Zustand als unangenehm erlebt: das Individuum versucht wieder einen Zustand der Balance[153] zu erlangen. Dem Individuum stehen verschiedene Strategien zur Bewältigung bzw. Reduktion der erlebten Dissonanz zur Verfügung. Entscheidend ist für den Aspekt der Stabilität der Gruppe, daß das Gruppenmitglied dazu neigt, einmal getroffene Entscheidungen so weit als möglich beizubehalten.

Die Attraktivität des Freien Software-Modells wurde soeben thematisiert (s.o.), sie stellt den Initial-Faktor für ein Engagement in Freier Software dar. Die Theorie der kognitiven Dissonanz wird verwendet, wenn diese initiale Phase, der Eintrit in eine Community, schon durchlaufen ist. Dieser Prozeß des Mitglied-werdens entspricht dem, was in der Theorie der kognitiven Dissonanz als Erfahrung oder als Entscheidung dargestellt ist. Somit versucht das Individuum, die eigene Existenz als Gruppenmitglied, was es als konsonanten Zustand empfindet, gegen dissonante Informationen oder Wahrnehmung aus der Umwelt der Gruppe in Einklang zu bringen. Der dissonante Zustand wirkt demnach motivierend, Handlungen zu ergreifen oder Verhaltensweisen anzunehmen, die die Dissonanz reduzieren. Dabei wird das Gruppenmitglied gegen jegliche Form von „dissonanter“ Information aus der Umwelt vorgehen, um seine Existenz als Gruppenmitglied, die Entscheidung dieser Gruppe anzugehören, vor sich und der Gruppe zu rechtfertigen. Dieses „Vorgehen“ ist – wie oben erwähnt – bestimmt durch den Versuch, ursprüngliche Entscheidungen beizubehalten, sie demnach auch zu verteidigen, wenn eine Übernahme einer neuen Information in den eigenen Erfahrungsschatz als „angemessen“ oder „objektiv richtig“ erscheinen würde. Dabei sind die Strategien, welche zur Erreichung eines möglichst konsonanten Zustands genutzt werden hier nicht von Relevanz – aber die Konsequenz für die Gruppe, für die Community. Ein derartiges Verhalten, welches sich mit der Theorie der kognitiven Dissonanz erklären läßt, schweißt die Gruppe weiter zusammen, der gemeinsame Versuch die gruppeninterne Konsonanz zu erhalten, stärkt das „Wir-Gefühl“.

Diese Spirale der zunehmenden „sozialen Zusammenschweißung“ der Gruppenmitglieder – unabhängig von der tatsächlichen oder der empfundenen Bedrohung durch die Umwelt oder dem Unverständnis von der Umwelt – wird solange das Fundament der Gruppe in ideologischer Hinsicht weiter fundieren, solange die Strategien zur Herstellung von Konsonanz greifen. Die Community erhält sich selbst.

Die Frage nach der Halbwertszeit einer Freien Software-Community stellt sich auch in Abgrenzung zu anderen sozialen Bewegungen[154]. Dabei fällt auf, daß sich die Ziele hierbei auf eine bestimmte Art und Weise unterscheiden können, und die Stabilität der jeweiligen Gruppe von der Formulierung dieses Ziels abhängig ist. Obwohl beispielsweise das Ziel der →Free Software Foundation und des →GNU-Projekts formuliert wurde, handelt es sich doch um etwas Abstraktes, teilweise Utopisches: Die Schaffung eines Freien →Betriebssystems und freier →Anwendersoftware, wie auch die Ablösung des proprietären Modells auf der Ebene der Software-Entwicklung und im Bereich des geistigen Eigentums generell, deutet eher auf den Prozeß zur Erreichung dieser „sehr fernen“ Ziele hin. Diese Unterscheidung der Prozeß-Orientierung versus Ziel-Orientierung ist mit verantwortlich für die Stabilität und lange Geschichte, auf die viele Freie Software-Projekte zurückblicken können. So liegt es in der Natur dieser Bewegung, daß sie ihr Ziel im Grunde nicht in absehbarer Zeit erfüllen kann – aber erstreben. Dabei hält das gemeinsame Streben, der gemeinsame Wille der Erfüllung eines fernen Zieles die Community – zumindest während des Strebens – zusammen. Das Gegenbeispiel wäre eine soziale Bewegung, welche sich ein konkretes Ziel, etwa das Verhindern einer Flughafen-Startbahn-Verlängerung, vorgenommen hat und mit Erreichen des Zieles wieder zerfällt, da der einzige gemeinsame Bezugspunkt wegfällt.

Ideologie

„Open Source, heißt es jetzt immer öfter, ist keine Produktionsmethode, kein Verfahren des freien Marktes. Es ist ein Weltbild. Eine Ideologie.“[155]

Freiheit

Da der Freiheitsbegriff in der vorliegenden Arbeit eine derart zentrale Stellung einnimmt, ist ihm hier ein eigener Punkt gewidmet, auch wenn er im Sinne des verwendeten Schemata der Aufzählung einzelner Motivationen nicht auf dieser Hierarchieebene einzuordnen ist. Freiheit, verstanden als Motivation für ein Engagement in Freier Software, fällt unter die Motivation, welche unter dem Oberbegriff „Ideologie“ in den folgenden Abschnitten formuliert ist.

Eine allgemeine Definition von Freiheit bezeichnet das Fehlen von inneren oder äußeren Zwängen. Sie impliziert damit Unabhängigkeit und Entscheidungsfreiheit.

Der hier im Zusammenhang mit Freier Software verwendete Freiheitsbegriff bezieht sich auf eine spezifische Form von Wissen – Software – als konkretes, immaterielles Gut. Die Verbindung von konkret und immateriell ist nötig, um den hier verwendeten Freiheitsbegriff gegenüber „ideellen“ Freiheiten wie etwa der Willensfreiheit abzugrenzen. Wenn bei Freier Software von Freiheit die Rede ist, dann bezeichnet diese immer eine konkrete Freiheit, eine Freiheit, welche direkt beobachtbar ist. Die Freiheit in Freier Software manifestiert sich in den Bestimmungen der Freien Software-Lizenz[156], in welcher hauptsächlich die folgenden vier Freiheiten postuliert werden:

- „The freedom run the program, for any purpose […].“

- „The freedom to study how the program works, and adapt it to your needs […].“
- „The freedom to redistribute copies so you can help your neighbor […].“
- „The freedom to improve the program, and release your improvements to the public, so that the whole community benefits […].“[157]

Auf der WOS I-Konferenz formuliert Richard Stallman, Begründer der genannten vier Freiheiten, die „Menschlichkeit“ dieser Freiheiten, indem er sagt:

„Außerdem sollte es möglich sein, eine Kopie [von einem Programm; Anm. d. Verf.] für einen Freund zu erstellen, so daß der Freund ebenfalls davon profitiert. Dies ist nicht nur nützlich, diese Art der Kooperation ist ein fundamentaler Akt von Freundschaft unter Leuten, die Computer benutzen. Der fundamentale Akt von Freundschaft unter denkenden Wesen besteht darin, einander etwas beizubringen und Wissen gemeinsam zu nutzen. […] Jedes Mal, wenn man die Kopie eines Programms weitergibt, ist dies nicht nur ein nützlicher Akt, sondern es hilft die Bande des guten Willens zu verstärken, die die Grundlage der Gesellschaft bilden und diese von der Wildnis unterscheiden. Dieser gute Wille, die Bereitschaft unserem Nächsten zu helfen, wann im-mer es im Bereich des Möglichen liegt, ist genau das, was die Gesellschaft zusammenhält und was sie lebenswert macht. Jede Politik oder jedes Rechtssystem, das diese Art der Kooperation verurteilt oder verbietet, ver-seucht die wichtigste Ressource der Gesellschaft. Es ist keine materielle Ressource, aber es ist dennoch eine äußerst wichtige Ressource.“ (Richard Stallman, in: Grassmuck 2002, S. 224)

Diese Freiheiten gehen nicht nur auf die Wertevorstellungen ihres Schöpfers – hier Richard Stallman – zurück, sie sind auch Ausdruck einer historischen Chance, der Trennung der Güter- von der Ideenwirtschaft[158] auch für und in der Software-Entwicklung[159] sowie den daraus entstandenen neuen Möglichkeiten von Produktion und Verwertung.

Setzt man den Freiheitsbegriff der Freien Software in Beziehung zur Freiheit in anderen Bereichen, so finden sich eine Reihe von Parallelen: Das Postulat der Gleichheit und der Transparenz sowie Elemente von Rede- und Meinungsfreiheit. Es sei damit an dieser Stelle nur angedeutet, daß die Grundprinzipien Freier Software sich in vielen Bereichen beispielsweise mit den Grundprinzipien der Demokratie decken. Im Rahmen dieser Arbeit von Interesse ist vielmehr die Beziehung zwischen der hier verwendeten Wissens-Freiheit anhand von Software und der Konstruktion von Eigentum: Freie Software versus proprietäre Software. Die Frage nach Eigentum, Nutzung und Verwertung von Wissen berührt auch die Frage nach den Rechten von „geistigen“ Leistungen einer Kultur. Um es mit Merton's Analogie[160] zu formulieren: Ist Software nicht von einer fähigen Minderheit aggregiertes Wissen einer Kultur? Software besteht aus Algorithmen und ihrer geschickten Verknüpfung – doch diese Algorithmen sind quasi der Schatz einer ganzen Kultur, sie sind ein öffentliches Gut, nicht das Eigentum eines einzelnen Software-Entwicklers. Der Software-Entwickler bedient sich für seine Arbeit aus diesem Pool an öffentlich verfügbarem Wissen, baut darauf auf und generiert neues Wissen. Ab diesem Punkt trennt sich die Freie und die proprietäre Software-Entwicklung: in proprietärer Software wird dieses „neue“ Wissen eingekapselt und für eine Gegenleistung – hier: monetäre Entlohnung – wieder an die Gesellschaft abgegeben; bei Freier Software dagegen wird auch dieses neue Wissen wieder in den Pool zurückgegeben, in der Hoffnung das öffentliche Gut zu vermehren, den Nutzen für alle Mitglieder dieser Kultur zu vergrößern und eine „öffentliche“ Evolution dieses Wissens zu ermöglichen. Auch wenn Robert K. Merton sein Bild zur Beschreibung der Wissensproduktion im

akademischen Bereich verwendete, ist die Analogie der Zwerge, die auf den Schultern von Riesen stehen und somit weiter sehen als der Riese selbst, auch heute, fast 40 Jahre nach Merton's Erstveröffentlichung[161] wieder aktuell, da diese Analogie eben jene Prozesse beschreibt, welche bei der Produktion von Freier Software involviert sind.

Das Entwicklungsmodell der Freien Software hat durch seinen Erfolg auch den Blick auf seinen Freiheitsbegriff gezogen. Als Konsequenz interessieren und engagieren sich bestimmte gesellschaftliche Gruppierungen für Freie Software. Beispielsweise die GRÜNEN, wenn sie sagen, daß „[d]er gesellschaftspolitische Hintergrund von OSS [Open Source Software; Anm. d. Verf.] berührt in vielfacher Hinsicht Grüne Werte. Partizipative und kooperative Wirtschaftsmodelle sind darin ebenso vertreten wie der freie Zugang zu Wissen.“ (Lichtenberger et al. 2004, S. 1). Auch im Bereich der Gewerkschaften, als Repräsentanten einer speziellen sozialen Bewegung – der Arbeiterbewegung – findet die Diskussion um die Grundfesten von Freier Software zunehmend Resonanz. Auf die Frage von ver.di[162] danach, ob Freie Software „etwas für Gewerkschaften ist“, antwortet Georg Greve[163]:

„Da Gewerkschaften Fragen wie Arbeitsbedingungen, gesellschaftliche Rahmenbedingungen und ähnlichen grundsätzlichen Fragen verpflichtet sind, die auch durch Freie Software angesprochen werden, würde ich meinen, daß es eigentlich nicht nur nahe liegend, sondern sogar fast verpflichtend ist, daß Gewerkschaften sich stark für Freie Software einsetzen.“ (Steinmann/ Meretz 2003, S. 2)

Des weiteren nimmt die Attraktivität der Prinzipien hinter Freier Software auch bei Künstlern immer mehr zu (siehe Kapitel 5.4.1.1), deren bisheriges Produktions- und Vertriebsmodell, wahrgenommen durch Verwertungsgesellschaften und auf Vervielfältigung ausgelegt war sowie durch den technischen Fortschritt, insbesondere der Möglichkeit der Digitalisierung ihrer Werke, bedroht ist.

Freiheit außerhalb Freier Software

Der Erfolg des Freien Software- und Open Source-Software-Entwicklungsmodells strahlt auch auf andere gesellschaftliche Bereiche ab. Nicht nur, daß eine neue Diskussion um geistiges Eigentum, Verwertungsrechte und informationelle Selbstbestimmung stattfindet, inzwischen wurden einige grundlegende Gedankten Freier Software auch in der Kunst[164] realisiert. Man kann vielleicht von der Renaissance des Urheberrechts sprechen – wenn nicht im gleichen Atemzug auch der Gegenwind – insbesondere aus den Patentämtern und der Patentindustrie – zugenommen hätte und sich momentan zu einer regelrechten Bedrohung[165] für dieses Entwicklungs- bzw. Produktionsmodell entwickelt.

Software und „klassische“ künstlerische Ausdrucksformen wie die Literatur oder die Musik haben technisch wie rechtlich Gemeinsamkeiten: Beide sind Ausdrucksformen der menschlichen Kreativität, die sich einem bestimmten Medium bedienen, beide fallen unter das Urheberrecht und gelangen so zu ihrem rechtlichen Schutz. So spricht zum Beispiel auch Werner Koch von seinen „Kollegen“ als „Software-Autoren“ (Koch 2004, S. 5)

Ein wichtiger Aspekt entsteht aber meines Erachtens durch die – bezüglich Software – extremen Bewertungsunterschiede hinsichtlich des „künstlerischen Ausdrucks“ und des reinen Gebrauchswertes. Was der Software-Entwickler als ein spielerisch-künstlerisches Projekt ausgibt, kann für einen Dritten – bzw. „sehr viele Dritte“ – einen sehr hohen Nutzwert, bis hin zu einem kommerzialisierbaren Wert erlangen. Diese Diskrepanz, vor allem der Nutzen von Software zur Be-

friedigung nicht-Lust-zentrierter Bedürfnisse wie beispielsweise die Unterhaltungsmusik, ist zu einem großen Teil mitverantwortlich an der gesamten Debatte um Software, Freiheit und Patentierbarkeit: die Universalität von Software zur Befriedigung verschiedenster Bedürfnisse, von Unterhaltung bis zur Steuerung von Kraftwerken, zieht Software unweigerlich in eine mächtige Verwertungsmaschine von Industrie und Wirtschaft ein.

Aber – um auf die Gemeinsamkeit zurückzukommen – so finden wir inzwischen Umsetzungen der Lizenzvergabe im Sinne des →Copylefts (siehe Kapitel 2.1.4 sowie 5.4.1.1) beispielsweise im Musik- und Film-Bereich:

Der vielleicht öffentlichkeitswirksamste Moment ist im November 2004 anzusiedeln, als das US-amerikanische IT-Magazin Wired eine Audio-CD mit namhaften Künstlern in einer Auflage von 750.000 Stück kostenlos unter das Volk brachte (Vgl.: Goetz 2004). Das „Neue“ an dieser CD ist, daß alle Stücke unter der Creative Commons License[166] stehen. Einer Lizenz, die es dem Künstler gestattet „[o]ffering your work under a Creative Commons license does not mean giving up your copyright. It means offering some [Hervorh. i. Orig.; Anm. d. Verf.] of your rights to any taker, and only on certain conditions.“[167] Die feinen Abstufungen in den Rechten, die der Künstler dem Lizenznehmer überschreibt, können so weit gehen, wie es bei der →GPL für den Bereich der Freien Software der Fall ist. Eben diese Möglichkeit wurde für die CD-Veröffentlichung beansprucht: „Copyleft – all rights reversed“ (Stallman 1999b, S. 59). Auch andere Künstler der selben Branche denken offen über neue Möglichkeiten der Verwertung ihrer Arbeit nach. So beispielsweise George Michael, der nach Bekanntgabe seiner Lizenzierungspläne zu verstehen gab, er sei „hopefully [...] a happier man, giving my music and also doing something really positive with my music.“ (BBC 3/2004).

In jüngster Vergangenheit wurden zudem zwei Spielfilme unter einer Creative Commons Lizenz veröffentlicht. Dazu gehört „Route 66 – Ein amerikanischer Traum“[168] sowie „CH7“[169].

Definition von Ideologie

Auf den Begriff der Ideologie wird in der vorliegenden Arbeit zurückgegriffen, um eine Motivation aus technologischen[170] Gründen von einer Motivation aus „weltanschaulichen“ Gründen abzugrenzen. Die Abgrenzung ist nicht absolut, sie deutet mehr auf die jeweils dominierende Betonung der entsprechenden Aspekte hin. Diese unterschiedliche Betonung findet man beispielsweise in der Arbeit und den Einstellungen um Linus Torvalds[171] und den Linux-Kernel – hier werden die technologische Überlegenheit und das vermeintlich bessere Entwicklungsmodell thematisiert – und den Mitstreitern des →GNU-Projekts unter der Ägide der →Free Software Foundation, für welche der herausragendste Aspekt der Gedanke der Freiheit in Software[172] ist. Die Grenze läßt sich grob zwischen den Vertretern und „Sympathisanten“ des Open Source- und des Freien Software-Lagers ziehen[173].

Damit sind zwei wichtige Begriffe der vorliegenden Arbeit – Ideologie und Freiheit – angesprochen. Aufgrund ihrer teilweise problematischen Verwendung, insbesondere der „Ideologie“ in den Sozialwissenschaften, soll im Folgenden eine vorläufige Definition für die Verwendung des Ideologe-Begriffs gegeben werden, der Begriff der Freiheit wurde in Kapitel 5.4.1 behandelt.

Im Rahmen der vorliegenden Arbeit wird Ideologie als eine Gesamtheit von Auffassungen und Ideen verstanden, die als direkte Grundlage für gesellschaftliche Veränderungen dienen. Diese Auffassungen sind zumeist einer spezifischen gesellschaftlichen Gruppierung zuzuordnen und beschreiben politische oder politische-theoretische Anschauungen. Der Ideologie-Begriff er-

scheint dem Autor besser als der allgemeine Begriff der „Einstellung“ geeignet, das Moment des Strebens, das Visionäre auszudrücken, auf die Kombination von Werten und Wünschen einzugehen, anstatt eine Zustandsbeschreibung zu liefern. Damit wird der Ideologie-Begriff hier als Überbegriff verwendet, und nicht als konkrete Auffassung von einer Ideologie. Diese Abstraktion hilft, die Wertbehaftung und oft problematische Ausdeutung des Begriffes[174] zu umgehen.

Die ideologische Motivation

Neben den bisher eingeführten Möglichkeiten der Kategorisierung der Mitglieder der Freien Software-Community (siehe Kapitel 5) besteht noch ein weiteres, vielleicht die für die Zukunft von Freier Software wichtigstes Unterscheidungsmerkmal: Der Grad der „ideologischen“ Motivation.

Historisch betrachtet stehen sich oftmals die ideologisch und die pragmatisch Motivierten gegenüber. Das pragmatisch motivierte Lager in der Freien Software-Community betont die technische Überlegenheit dieses Entwicklungsmodells und weist auf die oftmals diesbezüglich hinderliche Insistenz auf den ideologischen Faktoren hin. Das andere Lager setzt die Schwerpunkte in der unbedingten Verteidigung des Gedankens der Freiheit in Freier Software und sieht dies als ein Postulat an, welches nicht allein auf Software beschränkt ist oder sein sollte. Zu bemerken ist, daß dieses Lager keineswegs etwas gegen kommerzielle Verwendung von Freier Software einzuwenden hat[175], die durch die Freie Software-Lizenz garantierte Freiheit impliziert gerade auch die gleichberechtigte Nutzung sowie Entwicklung von Freier Software in einem kommerziellen Kontext.

Dieser Diskurs – die unterschiedliche Betonung des technologischen oder des ideologischen Aspektes – führte zu der vielleicht schicksalhaftesten Spaltung der beiden Lager im Jahr 1998, als sich bedeutende Mitglieder der Freien Software-Community unter dem Namen „Open Source“ neu formierten. Die Ambiguität des Begriffes „Free“, von der →Free Software Foundation verstanden im Sinne von „free as freedom, not as free beer“ (Vgl.: FSF 2004, S. 1), ihr anti-kommerzieller, fast sozialistischer Klang und das zunehmende Interesse der Wirtschaft, sollten so miteinander vereinbar gemacht werden, indem nicht mehr die Freiheit, sondern die Offenheit des →Quellcodes im Vordergrund steht. Der Begriff „Open Source“ referiert damit eher auf dem Organisationsprinzip, dem Entwicklungsmodell von Freier Software, als der Bedeutung einer allgemeinen Freiheit.

Die Geschichte hat gezeigt, daß auch „Open Source“ nicht vor Miß-Interpretationen gefeit ist, und wir können wieder eine Bewegung hin zur klassischen Definition von Freier Software feststellen. Mit der Einbringung eines neuen Begriffs war es nicht getan, auch wurden neue Lizenz-Modelle für Open Source Software entwickelt, beispielsweise die Mozilla Public Licence[176].

Wie der Name schon sagt, wurde und wird die „Offenheit“ des Quellcodes sehr betont. Für die klassische Freie Software ist ein offener Quellcode lediglich eine notwendige Bedingung, um die zweite [i. Orig.: „freedom 1“; Anm. d. Verf.] - und die vierte [i. Orig.: „freedom 3“; Anm. d. Verf.] Freiheit[177] zu ermöglichen.

Was aber bringt Menschen dazu, sich ideologisch für Freie Software einzusetzen, welche Aspekte können so wichtig sein, daß sie durch die Open Source-Bewegung nicht abgedeckt sind? Wir werden noch einige Motivationen kennengelernen, die alle einer Überprüfung auf „Pragmatismus“ standhalten und auch genügen, denken wir nur an „Spaß“ oder die Befriedigung von

„user needs“[178] im Sinne der Behebung von Mängeln (siehe Kapitel 5.6), auch eine Reihe der Community-basierten Aspekte lassen sich allein aus dieser Perspektive erklären.

Nun ist aber Software keine gesellschaftliche Komponente mehr, die sich als weitgehend abgekoppelt von der eigentlichen Entwicklung der Gesellschaft, quasi als ein Nebenschauplatz, darstellt[179]. Software wird mehr und mehr zu einem bestimmenden Faktor für Alltag, Arbeit, Beziehung, Wissen und Politik[180]. Schon aus dieser kurzen Aufzählung lassen sich einige Aspekte ableiten, die in direktem Zusammenhang mit Freier Software stehen: die Informations- oder Wissensgesellschaft, das Recht auf informationelle Selbstbestimmung, wie es die „verfassungsmäßige Ordnung“ (Schultzki-Haddouti 2003, S. 9) garantiert[181], oder Mechanismen der Zuweisung von Macht in einer Demokratie. Diese Punkte, sobald sie in irgendeiner Form über Software kanalisiert und verarbeitet werden, rufen förmlich nach einer Form der öffentlichen Kontrolle und nach Transparenz ihres Einsatzes.

Ein Beispiel für die Bedeutung von Freier Software wurde schon in Kapitel 3.1.2 anhand des US-amerikanischen Dilemmas mit elektronischen Wahlmaschinen angerissen und soll im Folgenden detailliert dargestellt werden. Auch wurden mögliche Konsequenzen einer „anderen“ Email-Kommunikation durchgespielt.

Freie Software – ein Moment demokratischer Kontrolle?

Marco Schulze formulierte mit fast prophetischer Note die aktuelle Diskussion um die Mißstände bei der US-amerikanischen Präsidentschaftswahl durch die Einführung von Wahlmaschinen (Vgl. Kapitel 3.1.2).

Wurde bisher von Freier Software als etwas berichtet, das sich in gewisser Weise „um sich selber dreht“, aus einer Sphäre der Arbeit von Freien Software-Entwicklern für Freie Software-Entwickler, so kann man spätestens hier einen Bruch ausmachen: Freie Software tangiert eine breite Öffentlichkeit, und die Diskussion um und Probleme mit proprietärer Software finden Raum auch in den für Normalverbraucher zugänglichen Medien[182].

Neben der tagesaktuellen Berichterstattung beschäftigt die Thematik inzwischen auch die Filmindustrie, die nach einigen Problemen mit der Veröffentlichung[183] Michael Moore's Film „Fahrenheit 9/11“ in die Kinos brachte, der sich – neben dem Hauptthema der Verbindungen der Familien Bush und Bin Laden – mit der Wahl George W. Bush's zum Präsidenten der Vereinigten Staaten von Amerika im Jahre 2001 und damit auch mit der „elektronischen Stimmabgabe“ beschäftigt[184]. Auch wenn einige Aussagen des Films angezweifelt werden[185], gewann der Film in den Kinos das Publikum für sich und in Cannes eine Goldene Palme. Auch staatliche Stellen[186] interessieren sich für die Vorkommnisse rund um die – damals – neu eingeführten elektronischen Wahlmaschinen. Die Journalistin Bev Harris hat sich der Thematik angenommen, ein Buch dazu publiziert[187] und unterhält eine Kampagnen-Webseite[188]. Aus aktuellem Anlaß sei auch auf einen Report der Organisation für Sicherheit und Zusammenarbeit in Europa (OSZE) zur US-Präsidentschaftswahl 2004 verwiesen[189].

> Aber was wird eigentlich bemängelt? Benutzen wir nicht ständig Computer, ohne Einblick in die Funktionsweise dieser Maschinen zu haben – bei einem Bankautomat zum Beispiel? Was die Brisanz der Wahlmaschinen ausmacht, ist, daß auch sie eine Art Computer sind, auf denen eine bestimmte Art von Software läuft und die eingehenden Daten – die „tatsächlichen“ Wählerstimmen – auf irgendeine Art und Weise transformiert und zu einem Ergebnis verarbeitet. Die Brisanz liegt in diesem „auf ir-

> gendeine Art und Weise“. Übertragen auf das konkrete Beispiel mit den Wahlmaschinen bedeutet das: Der Wähler gibt seine Stimme ab, mit dieser Stimmabgabe passiert etwas und am Ende dieser Verarbeitunskette – deren Länge nicht bekannt ist – wird dem Wähler das Wahlergebnis präsentiert. Nun ist es aber überhaupt nicht notwendig, daß der einzelne Wähler weiß, wie genau – aus technischer Perspektive – seine Stimme zum Wahlergebnis beiträgt. Wenn ein Bankautomat einen Fehler machen sollte, dann ist das für den aufmerksamen Bankkunden insofern transparent, daß ihm im jeweiligen Abrechnungszeitraum eine Kontroll- und Einspruchsmöglichkeit eingeräumt wird. Diese Möglichkeit der Kontrolle besteht so für den einzelnen Wähler nicht mehr – für den Wähler ist das Verfahren der Auszählung und Berechnung nicht transparent.

Entscheidend ist, daß es unabhängige Instanzen gibt, die über diese technische Prozedur Bescheid wissen sowie die korrekte Implementation der Auszählungsmechanismen kontrollieren. Für diese Kontrolle ist zwingend der Zugang zum →Quellcode der benutzten Software nötig – ebenso wie es die Free Software Definition für die Freiheit zwei[190] und vier[191] postuliert. Dies wiederum würde bedeuten, daß der Quellcode des „unter der Wahlsoftware laufenden“ →Betriebssystems ebenso zur Einsicht bereitstehen müsste wie die Wahlsoftware selber. Diese Trennung geht darauf zurück, daß es nur endlich viele →Betriebssysteme gibt und deren Entwicklung keine triviale Tätigkeit ist. Daher werden auch für z.B. Wahlmaschinen keine „eigenen“ →Betriebssysteme geschrieben, sondern es wird auf vorhandene zurückgegriffen. Das selbe passiert – fast – überall, wo Computer eingesetzt werden, ob das der schon genannte Bankautomat, die Flughafenorganisation oder ein Kernkraftwerk ist. Ein Großteil der elektronischen Prozesse, denen wir sichtbar oder unsichtbar in unserem Alltag begegnen, laufen über ein paar Dutzend →Betriebssysteme und eine jeweils sehr viel höhere Zahl von Spezialanwendungen. Eine korrekte Zertifizierung dieser Spezialsoftware – beispielsweise der Steuerungssoftware eines Kraftwerkes – wird auch von Laien als selbstverständlich betrachtet. Diese Einschätzung trifft – meines Erachtens – nicht für die jeweiligen →Betriebssysteme zu, obwohl dieses grundlegende Funktionen für ein korrektes Ablaufen der Spezialsoftware bereitstellt. Durch die monoplistischen, privat-wirtschaftlichen Strukturen auf dem Betriebssystemmarkt liegt faktisch die Kontrolle bei den Entwicklern und „Eigentümern“ der →Betriebssysteme – keineswegs beim jeweiligen Lizenznehmer, sei dies ein Elektrizitätsversorger oder die Bundesrepublik Deutschland. Es ist allerdings Realität, daß großen und/oder wichtige Kunden eines →Betriebssystems – und beide eben genannten Beispiele fallen darunter – ein Einblick in den Quellcode von proprietären Systemen gewährt wird[192]. Dieser Einblick ist allerdings niemals „perfekt“, sondern immer unvollständig. Und genau darin liegt das Problem: Die „Freiheiten“, wo und wie eine Funktion in Software eingebaut wird, sind fast unendlich groß. Daher können 99 % des Codes publiziert werden, doch in dem fehlenden einem Prozent steckt entweder die ganze Genialität oder die Schadensroutine dieser Software. Die Frage nach der Mutwilligkeit stellt sich hier überhaupt nicht, da es gesellschaftlich nicht relevant ist, aus welchem Grund eine eventuelle technische Fehlfunktion existent ist, sondern welche Konsequenzen diese hat. Somit ist für eine vollständige Überprüfung oder Kontrolle nur der komplette →Quellcode interessant[193]. Dies ist nur bei Freier oder Open Source Software möglich.

> Wir haben hier eine Verquickung von technischen und ideologischen Aspekten. Ist die Möglichkeit, den Quellcode einzusehen technischer Natur, so ist die Forderung nach dieser Möglichkeit primär ideologisch. Ein weiterer interessanter Punkt ist, daß immer mehr Menschen eine Offenlegung des Quellcodes wünschen, ohne jemals selbst die – technischen – Fähigkeiten und das Wissen zu haben, um damit über-

haupt etwas anzufangen, sie vertrauen dem öffentlichen Peer-Review-Prozeß, der sich an ihrer Stelle mit dem Quellcode befassen kann.

Exkurs: Utopie oder Wirklichkeit?

Aus aktuellem Anlaß zitiere ich eine Meldung vom 25. November 2004 des Heise-Verlages im Ganzen, da dies meines Erachtens der in dieser Arbeit bisher als „utopisch“ formulierten Zusammenhänge zwischen Software, Wahlmaschinen und Macht eben jenes „utopisch“ nimmt. Die Deutlichkeit der Nachricht und die erstmalige wissenschaftliche Untermauerung des Problems verbietet meines Erachtens die „Unterbringung“ in einer Fußnote.

„Wissenschaftler stützen These von Wahlbetrug bei US-Präsidentschaftswahl

Wissenschaftler der University of California in Berkeley haben eine statistische Analyse zu Unregelmäßigkeiten im Bundesstaat Florida bei der US-Präsidentschaftswahl durchgeführt. Die Forscher kommen zu dem Ergebnis, daß es einen eindeutigen Zusammenhang zwischen der Verwendung elektronischer Wahlmaschinen und einem überproportional hohen Stimmenanteil für George W. Bush gibt. Analysiert wurden unter anderem Faktoren wie Wahlbeteiligung, Wählerregistrierung, Wechselwählertrends und ethnische Herkunft der Einwohner. Die Wissenschaftler um Professor Michael Hout stellen folgende These auf: In allen Landkreisen, wo elektronische Wahlmaschinen zum Einsatz kamen, erhielt Bush - statistisch gesehen und bezogen auf die Zahl registrierter Demokraten - deutlich mehr Stimmen als eigentlich zu erwarten gewesen wären.

Die Berkeley-Wissenschaftler stützen damit Vorwürfe des US-Politikers Jeff Fisher, der schon kurz nach der Wahl auffällige Diskrepanzen beim Vergleich von Statistiken über Wählerregistrierungen und der Stimmenauszählung am 2. November in Gegenden, wo Wahlstimmen eingescannt wurden, ausgemacht hatte. In der Summe kommen Hout und seine Kollegen auf 130.000 respektive 260.000 irreguläre Stimmen -- je nachdem, ob es sich dabei um "Ghost-Votes" handelte, also Stimmen, die Bush einfach hinzugerechnet wurden, oder um tatsächlich abgegebene Stimmen, die vorsätzlich geändert wurden, mit dem Ergebnis, daß nicht nur Bush Stimmen mehr, sondern Kerry gleichzeitig die selbe Anzahl Stimmen weniger erhielt. Laut der Wahlbehörde Floridas hatte Bush den Staat mit einem Vorsprung von 380.000 Stimmen gewonnen.

Auffällig ist nach Angaben der Wissenschaftler auch, daß insbesondere dort, wo Demokraten traditionell ihre Hochburgen in Florida haben, Bush die meisten Extra-Stimmen erhielt. Allein für Broward County ermittelten die Forscher rund 76.000 "Bush-Stimmen", die "mit 99,9-prozentiger Sicherheit" nicht mit Veränderungsprozessen in der Bevölkerung erklärt werden können. Die Wissenschaftler sehen in den Zahlen vielmehr Anhaltspunkte dafür, daß entweder schon vor der Wahl die Wahlmaschinen mit Stimmen gefüttert wurden oder daß das Ergebnis nach dem Wahlgang durch Software-Manipulationen oder Hacker-Angriffe beeinflusst wurde.“ (Ziegler 2004)

Freie Software als Politikum

An dieser Stelle möchte ich noch einmal detaillierter auf die Bedeutung von Freier Software als ein Politikum eingehen.

Beginnen wir auf der technischen Ebene: Sprachen wir eben von Kanalisation und Verarbeitung von Informationen durch Software, geschieht dies in der Realität über irgendeine Form von

Vernetzung von Rechnern. Diese Rechner benutzen, je nach Form der angestrebten „Kommunikation“, unterschiedliche „Sprachen“ – oder →Protokolle, wie es technisch korrekt lautet. Auf beiden Seiten – Gesellschaft und Sprache sowie Internet und Protokolle – kommt es darauf an, daß diese Sprachen „frei“ sind, und Person A nicht finanziell oder sonstwie belastet werden kann, weil es Sprache B spricht. Die Universalität und die Freiheit von Sprache ermöglicht uns heute eine prinzipiell unbeschränkte Kommunikations-Chance. Ebenso müssen die Protokolle auf der Rechner-Ebene frei sein: Sie müssen von jedem Interessierten studiert und in eigene Projekte implementiert werden dürfen, ohne beispielsweise dafür Lizenzzahlungen leisten oder „Non-Disclosure-Agreements“ [dt.: Geheimhaltungsvereinbarung oder Vertraulichkeitsvereinbarung] unterzeichnen zu müssen, da Kosten und vor allem Einschränkungen an die Nutzer „vererbt“ werden. Die Lobby der Kulturschaffenden in Österreich[194] formuliert, vertreten durch Peter Riegersperger, diesen Umstand wie folgt:

„Im Gegensatz dazu [der Einsichtnahme von Großkunden in den Quellcode proprietärer Software; Anmerk. d. Verf.] ist Free Software mehr als ein Software Engineering Konzept: Das Credo von Free Software ist 'Information should be free'. Und tatsächlich handelt es sich bei Source Code ja um nichts anderes als Information.“ (Riegersperger 2001, S. 1)

Das →Debian-Projekt[195] sieht sich selbst als einen Vorreiter der Freiheit in Freier Software bei gleichzeitiger Balance mit den Interessen der Benutzer. Zur Erhaltung dieser Freiheit – des Debian GNU/Linux Systems – wurde 1997 der „Debian Social Contract“[196] kreiert, der später als Vorlage für die „Open Source Definition“[197] gelten wird.

Nach der Behandlung der Motivationen mit mindestens einer notwendigen sozialen Dimension, wird im Folgenden auf drei weitere wichtige Motivationen eingegangen. Dabei steht nicht die soziale Dimension im Vordergrund, welche bei Spaß, Mangel und Weiterbildung (s.u.) auch nicht notwendig vorhanden sein muß, sondern der Erklärbeitrag, den diese für das Gesamtphänomen Freie Software leisten. Sie sind im Rahmen dieser Arbeit aufgeführt, da zum Einen in ihnen oftmals der „initiale Moment“ für ein Engagement für Freie Software liegt, zum Anderen sie in den meisten Fällen die oben genannten Motivationen „begleiten“ und so zu einer zusätzlichen Varianzaufklärung beitragen. Weiter wäre jeder Versuch der Beschreibung von Freier Software ohne die folgenden drei Motivationen unvollständig.

Hedonismus

"Happy hacking!"

Abschiedsfloskel von Linus Torvalds

(Torvalds 1991, S. 5)

Auch wenn die pure Freude am Schreiben von Software keine „soziologische“ Motivation ist, da ihr bei alleinigem Auftreten nach der hier verwendeten Definition das Involviert-sein von Dritten fehlt (Vgl.: Einleitende Bemerkungen zu Kapitel 5), kann sie in dieser Arbeit nicht komplett fehlen, erklärt sie doch einen sehr großen Teil des Schaffens in Freier Software: 60 % der befragten Freien Software-Entwickler stimmen der Aussage „With one more hour in the day, I would program.“ mindestens mit „regelmäßig“ zu[198] (Lakhani et al. 2002, S. 11). In fast jeder weiteren vorliegenden Literatur finden sich Verweise auf das „Joy of Hacking“ (Raymond 2001b, S. 82). Eine sehr „starke“ Formulierung stammt von Donald Knuth[199], der behauptet „[…] daß

es möglich ist, großartige, edle, ja wahrhaft überwältigende [Hervorh. i. Orig.; Anm. d. Verf.] Programme zu schreiben!“ (Knuth in: Moody 2001, S. 217/218).

Wenn hier von „Joy“ oder „Fun“ die Rede ist, so involviert dies eine Reihe weiterer, eng verknüpfter Bereiche wie Befriedigung von Neugierde, Erlebnisorientierung, Kreativität oder Problemlösung allgemein. Auf die unzähligen Untersuchungen zu der „Freude des Hackens“ wird hier nicht weiter eingegangen und statt dessen auf das Forschungsprojekt „Fun and Software Development“ (FASD)[200], das einen Zusammenhang zwischen Spaß und dem Engagement in Freier Software herzustellen versucht, verwiesen. Wie auch bei anderen Studien zu „Spaß“ wird auf bewährte psychologische Theorien zurückgegriffen, hier das „Flow-Konzept“ von Csikszentmihalyi (Vgl.: Csikszentmihalyi 1985, S. 103ff, 191-249).

Aber was kann ein Fun-basiertes Motivationskonzept für die oben ausführlich behandelten Motivationen – Reputation, Altruismus, Community sowie Ideologie – an Zusatznutzen bringen?

Meines Erachtens wird dem simultanen Auftreten der verschiedenen Motivation nicht genügend Aufmerksamkeit gezollt. So fallen – je nach Kategorisierung – einige mögliche Motivationen unter den Tisch, während andere überbewertet werden. Denn um dem Anspruch, eine allumfassende Erklärung für das Phänomen Freier Software aus motivations-theoretischer Sicht gerecht zu werden, reicht dieser Rahmen nicht aus. Wozu er aber im Stande ist, ist die Betonung der einzelnen Faktoren und des möglichen Zusammenspiels dieser. Während ich mich hauptsächlich auf die „soziologisch interessanten“ Motivationen gestützt habe, möchte ich an dieser Stelle darauf hinweisen, daß auch eine Motivation wie Spaß einen nicht unerheblichen Beitrag zur Erklärung der genannten Motivationen leistet. Es werden oft Einwände und Einschränkungen gegen die oben genannten Motivationen – Reputation, Altruismus, Community sowie Ideologie – vorgebracht, die alle ihre Berechtigung haben. Jedoch wird dabei oft vergessen, daß die Akteure auch bei Fehlen von den oben genannten Motivationen oder Rückschlägen, was die Befriedigung dieser anbelangt, immer noch empfinden, daß „es unglaublichen Spaß macht [zu programmieren; Anm. d. Verf.]“ (Welte 2004, S. 10). Es können demnach lange „Durststrecken“ überwunden werden, denn Software-Entwicklung selbst wird von vielen als „eine sehr befriedigende Tätigkeit“ (Schulze 2004, S. 3) empfunden. Aus psychologischer Perspektiven ist die Freude an einer Tätigkeit an sich das Beispiel einer intrinsischen Motivation und kann als alleinige Erklärung für die Wahl einer bestimmten Handlungsalternative herhalten. Auch wenn wir trotz der Stärke des Motivs Spaß kaum ein isoliertes Auftreten von diesem beobachten können, tritt es bei fast allen Akteuren parallel zu anderen Motivationen auf und kann somit als Erklärung „einspringen“, wenn andere Motivationen versagen oder für sich genommen zu wenig Varianz aufklären.

Mangel

„Linux habe ich 1994 entdeckt […] auf der Suche nach der Lösung von meinem Problem.“ (Welte 2004, S. 1) Auch der Open Source „Evangelist“ (Vgl. Kapitel 5.3) Eric Raymond führt dies als Initialzündung von Freien Software-Projekten an: „Every good work of software starts by scratching a developer's personal itch.“ (Raymond 2001a, S. 4f).

Dieses Problem, von dem Harald Welte spricht, bezeichnet einen Mangel – einen Mangel, der technischer oder ideologischer Natur sein kann.

Der Mangel als Motivation aus technischer Perspektive

Wenn in der Literatur von „to scratch an itch“ (ebd.) die Rede ist, dann bezieht sich der Autor meist auf einen technischen Mangel: Im Falle Richard Stallman's war es der fehlerhafte Druckertreiber[201], bei Marco Schulze's Geschäftskompagnon der defekte Speicher[202], bei Eric Raymond der fehlende →Email-Client[203] – um nur drei Beispiele zu nennen. Dabei ist es unerheblich, ob dieses Problem als Erkenntnis eines Mangels bei der Nutzung von Freier Software wie bei Richard Stallman auftritt, oder ob beispielsweise ein bestimmtes →Betriebssystem mit defekten Arbeitsspeicher nicht zurecht kommt und auf ein anderes System ausgewichen werden muß, wie es im Interview von Marco Schulze zur Sprache kam Oder aber ob eine Funktionalität gänzlich fehlt, die dann – unabhängig ob in Freier oder auf andere Art und Weise lizenzierter Software – von demjenigen implementiert wird, der, wie Eric Raymond, diesen Mangel empfindet. In jedem Fall hat es die Menschen dazu gebracht, ihren Teil an Freier Software beizutragen.

Zwei Einschränkungen sind hierbei zu beachten: zum Einen müssen diese Menschen über das nötige Wissen verfügen, um ein benötigtes Stück Software selbst zu entwickeln, zum Anderen müssen sie insoweit von Freier Software „überzeugt“ sein, daß sie einen beispielsweise unter →Linux nicht perfekt laufenden Druckertreiber als Mangel empfinden – obwohl eventuell für andere →Betriebssysteme eine funktionierende Lösung bereitsteht.

Diese Motivation, insbesondere von ihrer technischen Seite, fällt gewissermaßen aus dem Rahmen, den die anderen Motivationen aufspannen: Er – der Mangel als Motivation – kann sich als einfacher Sachzwang darstellen, und dessen Eliminierung weder mit Spaß noch mit einer Lust am Problemlösen zusammenhängen muß. Der Mangel an sich muß daher keineswegs in die Freie Software führen. Nur sind all die Mängel, welche durch betroffene Software-Entwickler behoben wurden und die nicht in eine Freie Software-Laufbahn mündeten schlichtweg nicht zu quantifizieren. Für das empirische Auge sind sie unsichtbar, wir hören und lesen nur von den Mängeln, welche am Anfang – sehr oft als Anfang – einer Freien Software-Laufbahn stehen.

Die Empfindung dieses technischen Mangels kann sehr unterschiedlich ausfallen. Erwähnt wurde schon eine fehlerhafte Software, aber es sind weitere „Mängel“ denkbar: Der persönliche Anspruch kann Mängel projizieren, wie dies im Falle fehlender Dokumentation geschieht. Diese fehlende „Nebenprodukt“ von Software hat auch Martin Michlmayr in die Freie Software-Community „geschleust“:

„Und da habe ich dann eben mitgemacht [bei einem Freien Software-Projekt, hier: gnustep; Anm. d. Verf.] […] Aber ich bin kein Programmierer – ich habe immer Koordination gemacht. Bei gnustep habe ich angefangen Dokumentation, FAQ's zu schreiben, weil's damals einfach sowas nicht gegeben hat.“ (Michlmayr 2004, S. 2)

Man findet in beinahe jeder Literatur, die sich mit der Motivation um und in Freier Software beschäftigt, mindestens einen Hinweis auf den Mangel als Motivation, ebenso taucht der Mangel als Motivation in sämtlichen Interviews auf. Exemplarisch sei hier nur Werner Koch erwähnt, in dessen Freier Software-“Biographie“ auftaucht, daß: „Zufällig hat es sich mal ergeben, daß ich mir einen Laptop gekauft hatte, dazu eine Ethernet-Karte [Netzwerk-Karte; Anm. d. Verf.] und als ich GNU/Linux darauf installieren wollte, merkte ich: Hoppla, die Karte wird ja gar nicht unterstützt. Das war 1996. Wenn ich nicht aufpasse, was ich da kaufe, dann muß ich jetzt halt einen Treiber schreiben. Das war dann plötzlich möglich und ich habe den Treiber geschrieben.“ (Koch 2004, S. 2).

Der Mangel als Motivation aus ideologischer Perspektive

Auch aus ideologischer Perspektive kann ebenfalls ein Mangel empfunden werden: So können bestimmte Nutzungsbedingungen als inakzeptabel angesehen werden, wie dies sehr deutlich im Interview mit Harald Welte zur Sprache kam:

„Für mich persönlich muß ich sagen: Wenn es etwas nicht als Freie Software gibt, dann kann ich es eben nicht verwenden. Das ist halt eine sehr ideologische Entscheidung, ich lehne das einfach ab, das proprietäre Modell. […] Bis vor zwei Jahren konnte ich zum Beispiel meine Buchhaltung nicht machen unter Linux, weil's halt Gnucash [Freie Software für die private Buchhaltung; Anm. d. Verf.] noch nicht gab in der Form – dann mach' ich das halt nicht. Nicht mit Computerunterstützung, solange es diese Software nicht gibt. Entweder man schreibt sich dann diese Software, oder man ist bereit zu verzichten.“ (Welte 2004, S. 5f)

Diese Art des Mangels als Motivation wird unter Kapitel 5.4 behandelt.

Weiterbildung

Das Gefühl, eine Technik zu beherrschen, und diese Beherrschung zur Vollendung zu treiben, ist eine weitere wichtige Motivation für ein Engagement in Freier Software. Die starke Kategorie der „Beherrschung“ beherbergt hier auch das Bedürfnis nach einem – zumindest rudimentären – Verständnis von Alltagstechnik wie beispielsweise der Email-Kommunikation oder der Befriedigung einer Neugierde gegenüber den Möglichkeiten des Computers.

Die Ursachen eines derartigen Bedürfnisses können vielfältig sein. Man kann das positiven Gefühl der Beherrschung, das Selbstbildnis als Experten anführen, man kann auf die erwähnte allgemeine Neugierde zurückgreifen, oder auch die Hoffnung auf bessere Karrierechancen durch den Erfahrungsgewinn anführen.

Das Entstehen des Typus eines „Spezialisten“ (Popitz 1987, S. 640) beschreibt Popitz als den „erste[n] großen Differenzierungs-Schub der Gesellschaftsgeschichte (ebd.), also Folge der „Trennung von bäuerlichen und handwerklichen Tätigkeiten“[204] (ebd.). Doch die plötzlich entstandene Möglichkeit, die es dem Individuum erlaubte, sich nicht mehr um seine „universalen“ Belange selbst zu kümmern, sondern in einem bestimmten Gebiet – hier: dem Handwerk – aufzugehen, hatte nicht nur Auswirkungen auf neue berufliche Tätigkeitsfelder. Diese Möglichkeit schuf auch eine neue Quelle für die Befriedigung des Bedürfnisses nach Anerkennung und ein neues Selbstwertgefühl: der „Anerkennung in einer erworbenen Rolle“ (ebd.).

Auch in diesem Punkt führt die Motivation des Bedürfnisses der Weiterbildung über das Erwerben von Expertenwissen beispielsweise zu dem Bedürfnis, dieses Expertenwissen zu nutzen, um ein Signal an mögliche Instanzen, welche dieses Wissen honorieren und würdigen, auszusenden. Dies können Entscheidungsträger in der traditionellen Software-Entwicklung, aber auch die Community, in welcher das Wissen erworben wurde, sein. In jedem Fall führt die ursprünglich vorhandene Motivation zu einer neuen Umwelt des Individuums und in den meisten Fällen zu einer neuen Konstellation und Gewichtung von weiteren Motivationen (s.u.).

Allerdings sind alle der genannten Motivationen – Beherrschung und Neugierde – nicht alleine und spezifisch auf Freie Software anwendbar. Sie können auf fast jede Hobby-ähnliche Tätigkeit angewandt werden. Interessant erscheint Freie Software als Fokus der aufgeführten Motivationen erst, wenn man ihre Tauglichkeit zur Befriedigung der entsprechenden Bedürfnisse herauss-

tellt. Unter dieser Perspektive erscheint Freie Software prädestiniert zur Befriedigung der genannten Bedürfnisse: Die verwendeten Techniken stehen jedem Interessierten offen, sie zu studieren, uneingeschränkt zu verwenden sowie eigene Anpassungen und Veränderungen vorzunehmen[205].

Hervorzuheben ist, daß ein persönliches Engagement in Freier Software aus dem Interesse der Weiterbildung nicht unbedingt in der Mitarbeit in einem Freien Software-Projekt im Verbund mit Dritten münden muß – jedoch sind die Chancen dafür sehr hoch, da eine Beschäftigung mit Freier Software fast zwangsläufig über den Austausch mit Dritten über das Internet stattfindet, der sich schnell derart intensiviert, daß man mehr und mehr ein Zugehörigkeitsgefühl für eine bestimmte Gruppe empfindet. Ab diesem Punkt können sich weitere Motivationen entwickeln[206], und Mechanismen auf das Individuum einwirken, welche die Position des Individuums in der Gruppe stabilisieren (siehe Kapitel 5.3.2).

Die Literatur sowie die Interviews bestätigen diese Aussagen[207].

Abschließende Betrachtungen

Rekapitulation

Zu Beginn der Arbeit wurden, nach der Darstellung der Relevanz der Thematik und deren empirischen Fundierung, zwei Dimensionen eröffnet: zum Einen wurde eine neue Art von verteilter, öffentlicher Zusammenarbeit vorgestellt, ein neues Produktionsmodell – nicht nur – für die Schaffung von qualitativ hochwertiger und gesellschaftlich „nützlicher“ Software sowie die Frage nach den Beweggründen einzelner Individuen oder Gruppen für ein derartiges Engagement aufgestellt; zum Anderen wurde nach der sozialen Dimension dieses Engagements gefragt. In den folgenden Kapiteln wurde auf diese Fragestellungen detailliert eingegangen. Es wurde Wert darauf gelegt, daß ein möglichst vollständiges Bild von Freier Software-Entwicklung und Freien Software-Entwicklern gezeichnet wird. Dieses Bild setzt sich zusammen aus dem historischen Kontext Freier Software-Entwicklung (Kapitel 2), der Darstellung der gegenwärtigen Situation sowie der sozialen Einbettung des einzelnen Entwicklers (Kapitel 3) und der Analyse potentieller Motivationen für ein Engagement in Freier Software (Kapitel 5).

Die Ausführungen wollen zu einer Sensibilisierung eines alternativen Entwicklungsmodells beitragen, indem sie die Sphäre der Freien Software beschreiben und entsprechend der Fragestellung auf bestimmte Faktoren näher eingehen. Es wurden bezüglich der Motivation und des Engagements in Freier Software Motivationsfaktoren herausgearbeitet sowie nach ihrer sozialen Dimension kategorisiert. Dabei wurden Aspekte wie das Bedürfnis nach Anerkennung, altruistische Verhaltensdispositionen, Zugehörigkeitsbedürfnisse und spezifische Wertevorstellungen diskutiert (Kapitel 5.1 bis 5.4). Weiter wurde, um dem Phänomen Freier Software gerecht zu werden, Motivationen wie Spaß, Mangel und Weiterbildung kurz vorgestellt (Kapitel 5.5 bis 5.7), da sie einen nicht zu unterschätzenden Beitrag zur Erklärung des Gesamtphänomens leisten.

Zentrale Ergebnisse

Die Suche nach Anerkennung für das eigene Schaffen wird in der Literatur fast durchgängig und unbestritten als einer der wichtigsten Faktoren für ein Engagement in Freier Software dargestellt und soll deshalb hier nicht mehr explizit aufgeführt. Vielmehr soll im Folgenden ein

spezifischer Satz von zentralen Ergebnissen dieser Arbeit vorgestellt und auf Erkenntnisse eingegangen werden, welche bisher nach Ansicht des Autors nicht adäquat behandelt wurden.

Heterogenität und Variabilität der Motivationen

Die Heterogenität der jeweiligen Motivationen und die Komplexität der Motivationsbündel in Freier Software sind besonders hervorzuheben. Das Verhalten des Individuums kann in den wenigsten Fällen mit einer einzelnen der oben aufgeführten Motivationen zufriedenstellend erklärt werden. Vielmehr trifft man auf eine ganz persönliche Zusammenstellung unterschiedlicher Motivationsfaktoren. Es kann davon ausgegangen werden – dies wird auch durch Erkenntnisse aus den Interviews sowie der persönlichen Erfahrung des Autors gestützt –, daß das Verhalten eines Freien Software-Entwicklers meist von fast allen der in dieser Arbeit aufgeführten Motivationen mitgesteuert wird. Diese Einsicht wird von der Tatsache gestützt, daß die aufgeführten Motivationen nicht exklusiv zueinander stehen – ganz im Gegenteil, sie lassen sich leicht miteinander kombinieren. Die wechselseitige Verstärkung der Motivationen ist eventuell für die oftmals beobachtete „starke“ Überzeugung bezüglich des Freien Software-Modells bei Freien Software-Entwicklern verantwortlich. Um dies noch einmal beispielhaft zu verdeutlichen, sei auf die idealen Voraussetzungen des offenen Community-Gedankens mit geringen Einstiegsbarrieren[208] für eine altruistische Verhaltensdisposition hingewiesen. In diesem Zusammenhang sind auch die Motivationen wie Spaß, Mangel und Weiterbildung zu sehen: Sie liefern unter dem Aspekt der Heterogenität einen großen Beitrag zur Erklärung des beobachteten Verhaltens bezüglich Freier Software-Produktion. Einen Beitrag, der auch wirksam das Verhalten weiter steuert, wenn Bedürfnisse der in dieser Arbeit hervorgehobenen Motivationen (Kapitel 5.1 bis 5.4) zeitweilig nicht befriedigt werden.

Die Beziehung der einzelnen Motivationen zueinander und ihre jeweilige Gewichtung unterliegen einer Veränderung über die Zeit. Diese Variabilität über die Zeit, ist nicht zufällig, sie folgt einem über große Teile der Freien Software-Gemeinde konsistenten Muster. Somit kann von klassischen Eintritts-Motivationen für ein Engagement in Freier Software gesprochen werden, welche im Laufe der Zeit für das betroffene Individuum eine immer geringere Gewichtung zugunsten von anderen, „höheren“ Motivationen erfahren. Betrachtet man einen Anwender von Freier Software, der für sich einen Mangel auf diesem Gebiet entdeckt – wie beispielsweise Harald Welte und das Fehlen eines →Protokolls für eine bestimmte Form der Email-Kommunikation[209] – so liegt in der Behebung dieses Mangels eine erste Motivation – hier das Programmieren des fehlenden Protokolls – für die Entwicklung Freier Software. Dieser „Einstieg“ in die Freie Software-Entwicklung ist zudem auch der Einstieg in ein gesamtes Entwicklungsmodell. Als Konsequenz ist meist nicht nur die Behebung des ursprünglichen Mangels zu beobachten, sondern auch eine im Zuge der Behebung zunehmende Einbindung in Gruppen-Prozesse des Freien Software-Entwicklungsmodells.

Damit können weitere Bedürfnisse, Chancen und Möglichkeiten einhergehen: die Erfahrung, daß auch andere Mitglieder dieser Gemeinschaft einen Nutzen an der eigenen Arbeit haben und dem →Kontributor für sein Schaffen Anerkennung zollen, das Gefühl des sozialen Eingebundenseins, sowie die Anreize, die durch die Teilnahme an diesem Entwicklungsmodell entsteht, das eigene Engagement über das ursprünglich intendiert Maß hinaus zu steigern.

Das angesprochene Muster besteht hierbei in der Entwicklung von „einfachen“ Motivationen (Kapitel 5.5 bis 5.7) wie Spaß, Mangel und Weiterbildung zu „höheren“ Motivation[210] wie Reputations-Suche, altruistischen Verhaltensdispositionen, dem Community-Gedanken sowie den ideologischen Motivationen (Kapitel 5.1 bis 5.4). Gestützt auf die durchgeführten Interviews

sowie den Darstellungen in der Literatur führt eine zunehmende aktive Beschäftigung bzw. Mitarbeit in Freier Software dazu, die grundlegenden Prinzipien dieses Arbeitens als wichtigen, als „zu verteidigenden“ Wertekanon wahrzunehmen und zu verinnerlichen.

Manifestation und Relevanz von ideologischer Motivation

Die in Kapitel 5.4 beschriebene ideologische Motivation stellt für den Fortbestand des Entwicklungsmodells von Freier Software einen entscheidenden Aspekt dar. Nur dadurch, daß Freie Software-Entwickler gegenüber den zu Grunde liegenden Wertevorstellungen und Prinzipien sensibilisiert sind, können und werden sie sich dafür einsetzen, daß eben diese Wertevorstellungen und Prinzipien erhalten und verteidigt werden. Es wurde festgestellt[211], daß der Freie Software-Entwickler für sich diese ideologische Perspektive im Laufe seiner Tätigkeit als Freier Software-Entwickler immer weiter verinnerlicht. Diese Übernahme der Konzeption des Freien Software-Modells durch das Subjekt schafft noch keinen gesellschaftlichen Rückhalt für dieses Modell. Aus diesen individuellen Tendenzen müssen sich kollektive, koordinierte Bestrebungen herausbilden und sich auf der Ebene des Kollektivs manifestieren. Man findet dieses Kollektiv heute auf nationaler wie internationaler Ebene in Form von Organisationen und Institutionen, teilweise fest an den jeweiligen behördlichen, staatlichen und privat-wirtschaftlichen Entscheidungsprozessen beteiligt, mit eigener Lobbyarbeit und Fürsprechern in der Politik[212]. Die Herausbildung dieser institutionalisierten Struktur für Freie Software spricht zum Einen für die Relevanz der Thematik und zum Anderen für den Primat der ideologischen Motivation über die weiteren in dieser Arbeit behandelten Motivationen: aus ideologischen Beweggründen haben sich Institutionen zum Schutz und zur Förderung von Freier Software herausgebildet – eine Entwicklung, wie sie keine der anderen Motivationen hervorgebracht hat. Ein Entwicklungsmodell, welches insbesondere mit dem Adjektiv „informell“ charakterisiert wurde, geht inzwischen weit über diesen informellen Entwurf hinaus.

Im vorigen Kapitel 6.2.1 wurde eine Unterscheidung aufgrund von „höheren“ und „einfachen“ Motivationen eingeführt. Dabei versteht man unter einer einfachen Motivation, daß allein die Verhaltensweisen und Tätigkeiten zur Erreichung der Befriedigung des auslösenden Bedürfnisses ein Befriedigungspotential besitzen, wie das beispielsweise bei Spaß der Fall ist. Demgegenüber involviert der Prozeß des Erreichens der Befriedigung einer höheren Motivation auch Verhaltensweisen und Tätigkeiten, die für sich genommen nicht motivierend, teilweise sogar demotivierend wirken können. Im Falle der hier behandelten ideologischen Motivation steht ein Zielzustand im Vordergrund, zu dessen Erreichung auch für Freie Software-Entwickler unbefriedigende Tätigkeiten – wie beispielsweise die Verfolgung von Urheberrechtsverletzungen gegenüber Dritten – gehören. Eine primär hedonistische Gruppe wäre nicht im Stande, diese Funktionen zu übernehmen. Auch wenn die Mehrheit der Freien Software-Entwickler angibt, daß Spaß und kreatives Arbeiten ihre wichtigsten Beweggründe für ihr Engagement in Freier Software sind, so sind sie doch auch bereit, aus dieser Perspektive unpopuläre Tätigkeiten, für sich und die Community zu übernehmen.

Die Community als Spielfeld für einen spezifischen Altruismus

Die Gruppe als wichtiges Element in der Freien Software-Entwicklung übernimmt nicht nur die Funktion der Befriedigung von Zugehörigkeitsbedürfnissen, sie stellt zudem eine notwendige Voraussetzung für das Wirken weiterer Motivationen dar. Eine funktionierende Community im Sinne von geteilten Wertevorstellungen und Erwartungen gegenüber anderen Gruppenmitgliedern stellt das Fundament für eine Zuweisung von Reputation, als auch für einen nachhaltig

attraktiven reziproken Altruismus. Wie schon erwähnt, soll der Reputations-Aspekt ausgeklammert, statt dessen die spezielle Beziehung zwischen Community und altruistischen Verhaltensdispositionen hervorgehoben werden. In Kapitel 5.2 wurde die spezifische Form des reziproken Altruismus als Erklärungsansatz für das Freie Software-Phänomen hergeleitet.

Das Entwicklungsmodell Freier Software erscheint als geeigneter Nährboden für altruistisches Verhalten, welches auf einer verzögerten, indirekten „Rückzahlung“ basiert. Die Transparenz des Entwicklungsprozesses und die jeweilige direkte Verknüpfung zwischen namentlicher Nennung des beitragsleistenden Individuums und dessen Beitrags, motivieren geradezu altruistisches Verhalten im Sinne der Definition des reziproken Altruismus. Unter dieser Perspektive läßt sich auch das Reputations-Motiv (Kapitel 5.1) als lediglich nachgelagerte Motivation einordnen, da die Zuschreibung von Reputation als reziproke Gegenleistung verstanden werden kann. Hierbei besteht allerdings das Problem, daß die angesprochene Hierarchie oder Abfolge der Motivationen nicht mit Sicherheit festgestellt werden kann. Dazu müßte eine Zuordnung von beobachtetem Verhalten als quasi endgültiger oder „um-zu-“ Motivation getroffen werden können. Diese Unterscheidung bezieht sich auf die mit einem erwünschten Zielzustand direkt zusammenhängenden Motivationen, sowie jenen, die mehr als Mittel zum Zweck, anstatt als persönlich reizvoll wahrgenommen werden.

Für den Autor stellt der Entwurf des reziproken Altruismus in Freier Software einen sehr guten und vor allem sehr flexiblen Ansatz zur Erklärung verschiedenster Aspekte des Phänomens Freier Software dar.

Ausblick

Im Kontext der vorliegenden Literatur und der teilweise sehr heftigen, emotionalen Debatte um Freie Software, versteht sich diese Arbeit als eine Momentaufnahme des lebhaften Diskurses um ein modernes Gruppen-, Organisations- und Gesellschaftsphänomens. Dabei wurde besonders auf die Verknüpfung der sozialen und gesellschaftlichen Aspekte und die Schaffung eines Überblicks über die Hintergründe dieses Phänomens, sowie der Herausstellung der Interdependenz verschiedener Bestimmungsfaktoren, Wert gelegt.

Vor diesem Hintergrund bieten sich weitere Fragestellungen sowie empirische Fundierung und Vertiefung einzelner in der vorliegenden Arbeit behandelten Fragen an. Von besonderem Interesse dabei ist der Entwurf des reziproken Altruismuses in Freier Software, als auch der ökonomische Aspekt, ausgedrückt durch die Gegenüberstellung der „proprietären“ und der „freien“ Produktionsfunktionen. Daran ließen sich weitere Untersuchungen von genuin gesellschaftsideologischer Bedeutung, wie der Idee des Wissenskommunismuses[213] und der damit einhergehenden Produktion und Wartung öffentlicher Güter und eine Überprüfung unter der Perspektive der Metapher der „Tragedy of the Commons“[214] anschließen. Generell könnte eine Diskussion um Fragen des geistigen Eigentums, öffentlicher Güter und der Verbindung von individuellem Engagement und möglicher Verwertungen dieses Engagements angestoßen werden, wie dies momentan auch im Bereich der Verwertung – beziehungsweise der Restriktion der Verwertung – wissenschaftlicher Publikationen und bei der immer weiter ausgeweiteten Patentierbarkeit der Fall ist.

Auch in diesem Zusammenhang interessant und ausbaufähig ist die Analogie der Freien Software-Entwicklung zu der Produktion von Wissen in wissenschaftlichen Disziplinen, verstanden als „Climbing on the Shoulders of Giants“ (Vgl.: Kapitel 5.1.1), was hier lediglich angerissen werden konnte.

Freie Software ist zu einem vitalen und bedeutendem Element auch außerhalb des IT-Sektors für eine ganze Reihe weiterer gesellschaftlicher Bereiche geworden. Freie Software hat ein alternatives Entwicklungsmodell, nicht nur für die Software-Entwicklung, vorgestellt, sondern auch den Beweis angetreten, daß ein Entwicklungsmodell, getragen von Freiwilligen und zusammengehalten durch einen sozialen, auf das Gemeinwohl ausgerichteten Wertekanons sowie der weitgehenden Abwesenheit des Gedankens der Konkurrenz, in der Lage ist, hochwertige Güter herzustellen.

Die vorliegende Arbeit leistet einen Beitrag zur Erschließung dieser Thematik in der Soziologie. Es bleibt zu hoffen, daß die – sicherlich zwischen den Zeilen erkennbare – Begeisterung des Autors für die Thematik beim Leser Interesse weckt, denn Freie Software wird auch in Zukunft die Software-Entwicklung nachhaltig prägen.

Literaturverzeichnis

- **Amabile, Teresa**: How To Kill Creativity. In: Harvard Business Review, Watertown (MA) 76/5 1998, Seite 77-89.
- **Amabile, Teresa**: Creativity Under The Gun. In: Harvard Business Review, Watertown (MA) 80/8 2002, Seite 52-61.
- **Axelrod, Robert M.**: The Evolution of Cooperation. Basic Books, New York 1984.
- **Barbera, Francesco**: Berkman Center rebuffed. Copyright case dismissed by Center Court. In: Harvard Law Record, Cambridge 9/109 1999, URL: http://www.law.harvard.edu/students/orgs/forum/nader1.pdf, 1 Seite, letzter Abruf: 15.11.2004.
- **BBC News**, World Edition, o.A.: George Michael shuns music industry. o.O., 3/2004, URL: http://news.bbc.co.uk/2/hi/entertainment/3499534.stm, 2 Seiten, letzter Abruf: 10.07.2004.
- **Becker, David**: Former Microsoft exec joins open-source project. In: CNET News 15/11 2004, URL: http://news.com.com/2102-1046_3-5453259.html?tag=st.util.print, 3 Seiten, letzter Abruf: 16.11.2004.
- **Benkler, Yochai**: Coase's Penguin, or, Linux and The Nature of the Firm. In: The Yale Law Journal, o.O., 112 2002, URL: http://www.yale.edu/yalelj/112/BenklerWEB.pdf, 79 Seiten, letzter Abruf: 15.11.2004.
- **Bezroukov, Nikolai**: A Second Look at the Cathedral and the Bazaar. In: First Monday, o.O., 4/12 1999, URL: http://www.firstmonday.dk/issues/issue4_12/bezroukov/, 28 Seiten, letzter Abruf: 28.11.2004.
- **Burkart, Roland**: Kommunikationswissenschaft. 2. Aufl., Böhlau, Wien 1995.
- **Butler, Brian et al.**: Community Effort in Online Groups: Who does the Work and Why? In: Leadership at a Distance, o.O., 2002, URL: http://opensource.mit.edu/papers/butler.pdf, 32 Seiten, letzter Abruf: 01.07.2004.

- **Ciffolilli, Andrea**: The economics of open source hijacking and the declining quality of digital information resources: A case for copyleft. In: First Monday, o.O., 9 2004, URL: http://www.firstmonday.dk/issues/issue9_9/ciffolilli/, 14 Seiten, letzter Abruf: 03.07.2004.
- **Crott, Helmut**: Soziale Interaktion und Gruppenprozesse. Kohlhammer, Stuttgart 1979.
- **Csikszentmihalyi, Mihaly**: Flow. Das Geheimnis des Glücks. Klett-Cotta, Stuttgart 1990.
- **DiBona, Chris et al.**: Open Sources: Voices of the Open Source Revolution. 1. Aufl., O'Reilly, Sebastopol (CA) 1999.
- **Drosdowski, Günther** (Hrsg.): Das Fremdwörterbuch. 5. Aufl., Dudenverlag, Mannheim 1990.
- **Economist** (o.A.): Unix's founding fathers. In: The Economist, o.O., 2004, URL: http://economist.com/science/tq/PrinterFriendly.cfm?Story_ID=2724348, 2 Seiten, letzter Abruf: 28.07.2004.
- **Evers, Adalbert**: Freiwilliges und bürgerschaftliches Engagement, Ehrenamt, Selbsthilfe und Bürgergesellschaft - Begriffe machen Politik. In: AWO-Sozialbericht 2001: Ehrenamt im Wandel. Zum Internationalen Jahr der Freiwilligen., o.O., 1994.
- **Fischermann, Thomas**: USA: Elektronische Wahlfälscher? In: Die ZEIT online., 40/2004, URL: http://zeus.zeit.de/text/2004/40/wahlmaschine, 2 Seiten, letzter Abruf: 12.11.2004.
- **Fischermann, Thomas**: Neue Maschinen, neues Ergebnis? In: Die ZEIT online, 45/2004, URL: http://zeus.zeit.de/text/2004/45/us_fischermann, 2 Seiten, letzter Abruf: 11.11.2004.
- **Friedman, Roger**: Moore's 'Fellowship' of the 'Fahrenheit'? In: FOX News, 01/06 2004, URL: http://www.foxnews.com/printer_friendly_story/0,3566,130833,00.html, 2 Seiten, letzter Abruf: 16.11.2004.
- **Free Software Foundation** (FSF), o.A.: Why 'Free Software' is better than 'Open Source'. Boston (MA), 2001 URL: http://www.gnu.org/philosophy/free-software-for-freedom.html, 5 Seiten, letzter Abruf: 12.08.2004.
- **Free Software Foundation** (FSF), o.A.: The Free Software Definition. Boston (MA) 2004, URL: http://www.gnu.org/philosophy/free-sw.html, 3 Seiten, letzter Abruf: 15.11.2004.
- **Free Software Foundation Europe** (FSFE) o.A.: o.T. URL: http://www.fsfeurope.org/index.de.html, 2 Seiten, letzter Abruf: 27.12.2004.
- **Gardiner, Mary**: HOWTO Pay for Free Software. o.O. 2003, URL: http://users.puzzling.org/users/mary/HOWTO/Free/, 14 Seiten, letzter Abruf: 12.10.2004.
- **Garstka, Hansjürgen**: Informationelle Selbstbestimmung und Datenschutz. Das Recht auf Privatsphäre. In: Bürgerrechte im Netz, Bundeszentrale für politische Bildung o.O., 2003.
- **Gates, Bill**: Open Letter to Hobbyists. o.O., 1976, URL: http://www.tranquileye.com/cyber/1976/gates_open_letter_to_hobbyists.html, 2 Seiten, letzter Abruf: 05.10.2004.

- **Ghosh, Rishab et al.**: Free/Libre and Open Source Software: Survey and Study. Survey of Developers. International Institute of Infonomics, University of Maastricht 2002, URL: http://floss.infonomics.nl/report/FLOSS_Final4.pdf, 64 Seiten, letzter Abruf: 11.10.2004.
- **Goetz, Thomas**: Sample the Future. In: Wired 12/11 2004, URL: http://wired.com/wired/archive/12.11/sample.html, 5 Seiten, letzter Abruf: 17.11.2004.
- **Goltzsch, Patrick**: Musikalische Viren. o.O., 1999, URL: http://www.heise.de/tp/r4/artikel/2/2711/1.html, 2 Seiten, letzter Abruf: 24.11.2004.
- **Grassmuck, Volker**: Freie Software. Zwischen Privat- und Gemeineigentum. Bundeszentrale für politische Bildung (bpb), Bonn 2002.
- **Hardin, Garret**: The Tragedy of the Commons. In: Science 162 1968, Seite 1243-1248.
- **Harris, Bev**: Black Box Voting. Vote Tampering in the 21th Century. Plan Nine Pub, o.O., 2003, URL: http://www.blackboxvoting.com/, letzter Abruf: 21.10.2004.
- **Healy, Kieran**: Altruism as an Organizational Problem: The case of organ procurement. In: American Sociological Review, Philadelphia, 69 2004, Seite 387-404.
- **Henson, Val**: HOWTO Encourage Women in Linux. o.O. 2002, URL: http://www.tldp.org/HOWTO/Encourage-Women-Linux-HOWTO/, letzter Abruf: 17.11.2004.
- **Hertel, Guido / Niedner, Sven / Herrmann, Stefanie**: Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors in the Linux Kernel. Universität Kiel 2003, 41 Seiten, letzter Abruf: 09.12.2004.
- **Herzberg, Frederick**: The motivation to work. Wiley, New York 1959.
- **Hillmann, Karl-Heinz**: Wörterbuch der Soziologie. Kröner, Stuttgart 1994.
- **Huber et al.**: Frauen in IT- und ausgewählten technischen Ausbildungen und Berufen in Baden-Württemberg. Akademie für Technikfolgenabschätzung in Baden-Württemberg, o.O. 2001, URL: http://www.ta-akademie.de/deutsch/bestellungen/textelk/ab213/ab213a.pdf, 94 Seiten, letzter Abruf: 27.11.2004.
- **Inglehart, Roland**: Culture shift in advanced industrial society. Princeton University Press, Princeton University Press, Oxford 1990.
- **Koch, Werner**: Interview. Juni 2004, URL: http://www.tombreit.de/vers2/sites/uni/, 13 Seiten.
- **Kohn, Alfie**: Studies Find Reward Often No Motivator. o.O. 1987, URL: http://www.fsf.org/philosophy/motivation.html, 4 Seiten, letzter Abruf: 26.09.2004.
- **Kopel, David**: Fifty-nine Deceits in Fahrenheit 911. URL: http://www.davekopel.org/terror/59Deceits.pdf, 4 Seiten, letzter Abruf: 22.10.2004.
- **Krempl, Stefan**: Microsoft will bei 'Shared Source' nachbessern. In: Heise online, Hannover 2002, URL: http://www.heise.de/newsticker/meldung/31394, 1 Seite, letzter Abruf: 02.10.2004.

- **Krishnamurthy, Sandeep**: Cave or Community?: An Empirical Examination of 100 Mature Open Source Projects. In: First Monday, o.O., 6/7 2002, URL: http://www.firstmonday.org/issues/issue7_6/krishnamurthy/, 8 Seiten, letzter Abruf: 22.07.2004.
- **Krüger, Alfred**: Der große E-Voting Beta-Test. In: Telepolis, Hannover 11/2004, URL: http://www.heise.de/tp/r4/artikel/18/18718/1.html, 2 Seiten, letzter Abruf: 11.11.2004.
- **Lakhani, Karim et al.**; The Boston Consulting Group (Hrsg.): The Boston Consulting Group Hacker Survey. o.O., 2002, URL: http://www.ostg.com/bcg/BCGHACKERSURVEY-0.73.pdf, 43 Seiten, letzter Abruf: 12.08.2004.
- **Lakhani, Karim / Wolf, Robert**: Why Hackers Do What They Do: Understanding Motivation Efforts in Free/Open Source Software Projects. MIT Sloan School of Management, Working Paper 4425-03 2003, URL: http://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID443040_code030911590.pdf, 28 Seiten, letzter Abruf: 10.06.2004.
- **Lancashire, David**: Code, Culture and Cash: The Fading Altruism of Open Source Development. In: First Monday, o.O., 6/12 2001, URL: http://firstmonday.org/issues/issue6_12/lancashire/, 29 Seiten, letzter Abruf: 08.06.2004.
- **Lerner, Josh / Tirole, Jean**: The Simple Economics of Open Source. National Bureau of Economic Research, Cambridge 2000, URL: http://www.nber.org/papers/w7600, 40 Seiten, letzter Abruf: 01.07.2004.
- **Levy, Steven**: Hackers. Heroes of the Computer Revolution. 2. Aufl., Penguin Books, New York, 1994.
- **Lichtenberger, Eva et al.**: Open Mind - Open Source - Open Europe. European Greens, o.O., 2004, URL: http://www.gruene-muenchen-stadtrat.de/oss/linux/index.htm, 1 Seite, letzter Abruf: 12.10.2004.
- **Lindenberg, Sigwart**: Intrinsic Motivation in a New Light. In: Kyklos, Frey, René (Hrsg.), Basel 54 2001, Seit 317-342.
- **Lotter, Wolf**: Die Diktatur des Volontariats. In: brand eins, Hamburg 7 2001, URL: http://www.brandeins.de/ximages/11731_024diedikt.pdf, 7 Seiten, letzter Abruf: 26.05.2004.
- **Luthiger, Benno**: Fun and Software Developemnt. FASD Study Provisional Results. Universität Zürich 2004, URL: http://www.isu.unizh.ch/fuehrung/blprojects/FASD/FASD_eval1.pdf, 22 Seiten, letzter Abruf: 13.11.2004.
- **Maslow, Abraham**: Motivation und Persönlichkeit. Olten, Freiburg, 1978.
- **McHugh, Josh**: For the Love of Hacking. In: Forbes Magazin, o.O., 8 1998, URL: http://www.forbes.com/forbes/1998/0810/6203094a_print.html, 4 Seiten, letzter Abruf: 03.08.2004.
- **McKusick, Kirk**: Twenty Years of Berkeley Unix: From AT&T-Owned to Freely Redistributable. In: Open Sources: Voices of the Open Source Revolution, Sebastapol (CA) 1999, Seite 31-46.
- **Merton, Robert K.**: The Normative Structure of Science. In: Robert K. Merton, The Sociology of Science, University of Chicago Press 1974.

- **Merton, Robert K.**: Auf den Schultern von Riesen: Ein Leitfaden durch das Labyrinth der Gelehrsamkeit. Syndikat, Frankfurt a. M. 1980.
- **Michlmayr, Martin**: Interview. Juni 2004, URL: http://www.tombreit.de/vers2/sites/uni/, 13Seiten.
- **Moglen, Eben**: Interview: Eben Moglen über die Zukunft der GPL (Teil 1). o.O., 2004, URL: http://www.golem.de/0406/31795.html, 4 Seiten, letzter Abruf: 22.07.2004.
- **Möller, Erik**: Streit der Kulturen. Die Reformation zum Anfassen: GNU/Linux und Open Source - Teil 2. In: Telepolis, Hannover 2001, URL: http://www.heise.de/tp/deutsch/inhalt/te/9851/1.html, 12 Seiten, letzter Abruf: 12.07.2004.
- **Möller, Ingrid**: Frauen in Zukunftsberufen. Chance Multimedia. Senatsamt für die Gleichstellung, Hamburg, Hamburg, 2002, URL: http://fhh.hamburg.de/stadt/Aktuell/behoerden/gleichstellung/service/publikationen/frauen-in-zukunftsberufen-pdf,property=source.pdf, 200 Seiten, letzter Abruf: 27.11.2004.
- **Möller, Erik**: Die heimliche Medienrevolution - Wie Weblogs, Wikis und freie Software die Welt verändern. Heise, Hannover 2004.
- **Moody, Glenn**: Die Software Rebellen. moderne industrie, Landsberg/Lech 2001.
- **Moon, Jae Yun / Sproull, Lee**: Essence of Distributed Work: The Case of the Linux Kernel. In: First Monday, o.O. 5/11 2000, 15 Seiten, letzter Abruf: 01.07.2004.
- **Orr, Shepley**: The Economics of Shame in Work Groups: How Mutual Monitoring Can Decrease Cooperation in Teams. In: Kyklos, Frey, René (Hrsg.), Basel 54/1 2001, 18 Seiten.
- **Osterloh, Margit / Rota, Sandra / Kuster, Bernhard**: Open Source Software Production: Climbing on the Shoulders of Giants. Institute for Research in Business Administration, University of Zürich, 2002, URL: http://opensource.mit.edu/papers/osterlohrotakuster.pdf, 34 Seiten, letzter Abruf: 28.05.2004.
- **Pitzke, Marc**: Der Bezirk mit 139 Prozent Wahlbeteiligung. Zweifel am US-Wahlergebnis. In: Spiegel online, o.O. 11/2004, URL: http://www.spiegel.de/politik/ausland/0,1518,327359,00.html, 3 Seiten, letzter Abruf: 16.11.2004.
- **Popitz, Heinrich**; Neidhardt, Lepsius, Esser (Hrsg.): Autoritätsbedürfnisse. Der Wandel der sozialen Subjektivitvität. In: Kölner Zeitschrift für Soziologie und Sozialpsychologie, Opladen 39 1987.
- **Prüfer, Jens**: Why do developers and firms contribute to the production of Open Source Software? Universität Frankfurt/Main, 2004, URL: http://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID590581_code373723.pdf?abstractid=590581&mirid=1, 30 Seiten, letzter Abruf: 17.11.2004.
- **Raven, Alder**: Feminist Poet Crypto Geek. o.O., 2004, URL: http://www.oneeyedcrow.net/securitygeekfemme.html, 4 Seiten, letzter Abruf: 01.12.2004.
- **Raymond, Eric**: A Brief History of Hackerdom. In: Open Sources: Voices from the Open Source Revolution, O'Reilly, Sebastapol (CA) 1999, Seite 19–30.

- **Raymond, Eric** [2001a]: The Cathedral and the Bazaar. In: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary., 2. Aufl., O'Reilly, Sebastopol (CA) 1999, S. 19-64.
- **Raymond, Eric** [2001b]: Homesteading the Noosphere. In: The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary., 2. Aufl., O'Reilly, Sebastopol (CA) 2001, Seite 65-112.
- **Rheinberg, Falko**; Selg, Herbert / Ulrich, Dieter (Hrsg.): Motivation. In: Grundriss der Psychologie: Eine Reihe in 22 Bänden., 4. Aufl., Kohlhammer, Stuttgart 2002.
- **Richter, Steffen**: Patente auf Software? In: Die ZEIT, 30 2003, URL: http://www.zeit.de/2003/35/patent, 4 Seiten, letzter Abruf: 09.07.2004.
- **Riegersperger, Peter**; IG Kultur Österreich: Kultureller Stellenwert und politische Bedeutung von Open Source und freier Software. In: Kulturrisse, o.O., 01/02 2001, URL: http://www.igkultur.at/igkultur/kulturrisse/1011686765/1011691437, 2 Seiten, letzter Abruf: 13.05.2004.
- **Rivlin, Gary**: Leader of the Free World. In: Wired, o.O., 11/11 2003, URL: http://www.wired.com/wired/archive/11.11/linus.html, 9 Seiten, letzter Abruf: 16.10.2004.
- **Rossi, Maria Alessandra**: Decoding the Free/Open Source (F/OSS) Software Puzzle. A Survey of theoretical and empirical Contributions. Department of Economics, University of Siena, 424 2004, URL: http://www.econ-pol.unisi.it/quaderni/424.pdf, 42 Seiten, letzter Abruf: 28.05.2004.
- **Roth, Wolf-Dieter**: USA: Die Qual der Wahl. In: Telepolis, Hannover 10/2004, URL: http://www.heise.de/tp/r4/artikel/18/18666/1.html, 2 Seiten, letzter Abruf: 08.09.2004.
- **Rötzer, Florian**: Das Problem mit den elektronischen Wahlsystemen und der amerikanischen Demokratie. In: Telepolis, Hannover 7/2003, URL: http://www.heise.de/tp/r4/artikel/15/15193/1.html, 3 Seiten, letzter Abruf: 17.11.2004.
- **Rudolph, Udo**: Motivationspsychologie. 1. Aufl., Beltz, Weinheim 2003.
- **Ryan, Richard / Deci, Edward**: Self-determination theory and the faciliation of intrinsic motivation, social development, and well-being. In: American Psychologist, University of Rochester, o.O. 55 2000, 11 Seiten, letzter Abruf: 17.11.2004.
- **Schulze, Marco**: Interview. Freiburg, 2004, URL: http://www.tombreit.de/vers2/sites/uni/, 19 Seiten.
- **Schulzki-Haddouti, Christiane**: Bürgerrechte im Netz. Schriftenreihe Bd. 382, Bundeszentrale für politische Bildung, Bonn, 2003.
- **Stallman, Richard**: New UNIX Implementation. o.O., 1983, URL: http://www.gnu.org/gnu/initial-announcement.html, 3 Seiten, letzter Abruf: 04.08.2004.
- **Stallman, Richard**: The GNU Manifesto. o.O., 1985, URL: http://www.gnu.org/gnu/manifesto.html, 6 Seiten, letzter Abruf: 24.07.2004.

- **Stallman, Richard** [1999a]: Free Software And Beyond. Rede auf WOS-I, Berlin 1999, URL: http://wizards-of-os.org/index.php?id=502, 4 Seiten, letzter Abruf: 18.10.2004.
- **Stallman, Richard** [1999b]: The GNU Operating System and the Free Software Movement. In: Open Sources: Voices from the Open Source Revolution., 1. Aufl, O'Reilly, Sebastopol (CA) 1999.
- **Stangl, Werner**: Motiv und Motivation. Psychologische Erklärungsmodelle. Linz 2004, URL: http://www.stangl-taller.at/ARBEITSBLAETTER/MOTIVATION/MotivationModelle.shtml#Modell%20der%20Leistungsmotivation, 9 Seiten, letzter Abruf: 12.08.2004.
- **Steinmann, Bernd / Meretz, Stefan**: Software zum glücklich werden. ver.di-Interview mit Georg Greve.", o.O., 2003, 4 Seiten, letzter Abruf: 13.10.2004.
- **Sueddeutsche.de**/AP/dpa (o.A.) [5/2003]: Entscheidung im Stadtrat: München setzt auf Linux. München, 2003, URL: http://www.sueddeutsche.de/muenchen/artikel/128/12116/, 2 Seiten, letzter Abruf: 27.07.2004.
- **Torvalds, Linus**: Free minix-like kernel sources for 386-AT. o.O., 1991, URL: http://listserv.nodak.edu/scripts/wa.exe?A2=ind9110a&L=minix-l&F=&S=&P=8303, 2 Seiten, letzter Abruf: 22.07.2004.
- **Torvalds, Linus**: Re: SCO: 'thread creation is about a thousand times faster than onnative'. o.O., 2000, URL: http://lkml.org/lkml/2000/8/25/132, 1 Seite, letzter Abruf: 16.10.2004.
- **Torvalds, Linus / Diamond, David**: Just for Fun. Wie ein Freak die Computerwelt revolutionierte. Hanser, München 2001.
- **Tuomi, Ilkka**: Evolution of the Linux Credit file: Methodological challenges and reference data for Open Source. in: First Monday, o.O., 9/6 2004, URL: http://www.firstmonday.org/issues/issue9_6/tuomi/, 26 Seiten, letzter Abruf: 08.06.2004.
- **Unger, Bettina**: Wonder Women in the Rude Boys' Paradise? Wien 2003, URL: http://members.nusurf.at/bettina.unger/downloads/diplomarbeit_unger.pdf, 168 Seiten, letzter Abruf: 27.11.2004.
- **Weber, Max**; Johannes Winckelmann (Hrsg.): Die protestantische Ethik I. Siebenstern, München 1969.
- **Weiner, Bernhard**: Theorien der Motivation. Klett, Stuttgart 1976.
- **Welte, Harald**: Interview. Juni 2004, URL: http://www.tombreit.de/vers2/sites/uni/, 11 Seiten.
- **West, Joel / Dedrick, Jason**: Proprietary vs. Open Standards in the Network Era: An Examination of the Linux Phenomenon. Center for Research on Information Technologies and Organizations, University of California, Irvine 2001, URL: http://csdl.computer.org/comp/proceedings/hicss/2001/0981/05/09815011.pdf, 10 Seiten, letzter Abruf: 18.05.2004.
- **Wichmann, Thorsten**: Free/Libre and Open Source Software: Survey and Study. Use of Open Source Software in Firms and Public Institutions. Evidence from Germany, Sweden

and UK. Berlecon Research, Berlin 2002, URL: http://floss.infonomics.nl/report/report-Part1_use_oss_in_firms_and_public_institutions.htm, 64 Seiten, letzter Abruf: 11.10.2004.

- **Williams, Sam**: Free as in Freedom. Richard Stallman's Crusade for Free Software. O'Reilly, Sebastopol (CA), 2002.
- **Zetter, Kim** [10/2004] : Activists Find More E-Vote Flaws. In: Wired, o.O 2004, URL: http://www.wired.com/news/print/0,1294,65535,00.html, in: Wired, 4 Seiten, letzter Abruf: 17.11.2004.
- **Zettler, Kim** [3/2004] : How E-Voting Threatens Democracy. In: Wired, o.O. 2004, URL: http://www.wired.com/news/print/0,1294,62790,00.html, in: Wired, 15 Seiten, letzter Abruf: 17.11.2004.
- **Ziegler, Peter-Michael**: Wissenschaftler stützen These von Wahlbetrug bei US-Präsidentschaftswahl. Heise online, Hannover 2004, URL: http://www.heise.de/newsticker/meldung/53464, 1 Seite, letzter Abruf: 25.11.2004.

Anhang I – Glossar

Advertising Clause – „All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.“ (aus der Original-BSD-Lizenz, siehe: ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change)

AI-Lab – MIT Artificial Intelligence Laboratory. Ein interdisziplinäres Forschungszentrum am Massachusetts Institute of Technology, das 1959 von den Professoren Marvin Minsky und John McCarthy sowie einer Gruppe sehr engagierter Programmierer gegründet wurde.

Anwendungsprogramm – Software, welche nicht direkt dem Funktionieren des Computers dient, sondern zur Erledigung von Aufgaben der Nutzer gedacht ist, wie beispielsweise Textverarbeitung oder Tabellenkalkulation.

Anwendungssoftware – siehe →Anwendungsprogramm

ARPANET – Advanced Research Projects Agency Network. Computer-Netzwerk des US-Verteidigungsministeriums, welches erstmals Technologien wie die Paket-vermittelte Kommunikation (siehe auch: TCP/IP) einsetzte. Vorgänger des Internet.

Assembler – siehe →Assemblercode

Assemblercode – Eine Programmiersprache, die den spezifischen Befehlssatz einer Prozessorarchitektur in einer Menschen-lesbaren Form darstellt. Jede Prozessorachitektur hat somit seine eigene „Assemblersprache“ oder ihren eigenen „Assemblercode“

Bandbreite – Die Bandbreite gibt an, welche Datenmengen in einem bestimmten Zeitabschnitt über eine bestimmte Route geleitet werden können.

Berkeley Software Distribution (BSD) – Eine Version von UNIX, die von der Universität von Berkeley, Kalifornien, betreut und vertreiben wird. Es haben sich drei Entwicklungsrichtungen bis heute gehalten: OpenBSD, FreeBSD und NetBSD, die alle auf dem selben Code-Stamm aufbauten, jedoch unterschiedliche Schwerpunkte für „ihre“ Distribution wählten.

Betriebssystem – Die System-Software, die sich um die grundlegenden Funktionen eines Computers kümmert und die direkte Kontrolle über die Hardware besitzt. Das Betriebssystem ist die Grundlage für →Anwendungssoftware wie beispielsweise einem Webbrowser oder einer Textverarbeitung.

Binärversion – Binärversionen von Software entsprechen dem Objektcode des Programms, im Gegensatz zu dem Quellcode. Binärversionen gelten als nicht les- oder „verstehbar“ für Menschen, sie repräsentieren die Sprache der Computer-Plattform, auf dem die entsprechende Binärversion ausgeführt werden soll. Das Prinzip Freie Software funktioniert ausschließlich, wenn der Quellcode eines Programmes verfügbar ist, da nur durch das Studium und die Änderbarkeit des Quellcodes eine Modifikation des Programms möglich ist.

Binary – siehe →Binärversion

BSD – siehe →Berkeley Software Distribution

Bug/Bugs/Bug-Fixing – Als Bug [dt: Käfer] wird ein Fehler in Hard- oder Software bezeichnet. Der seltsam anmutende Name rührt von einer Motte her, welche 1945 in einem der ersten Computer zu einem Kurzschluß geführt hat. Heute bezeichnet man alle möglichen Arten von Fehlern in Programmen als „Bugs“, und das sogenannte „Bug-Fixing“ umschreibt die Arbeit, die Entwickler oder Unterstützer leisten um Bugs in einer Software zu lösen („to fix“). Diese Arbeit kann zu einem wesentlichen und nicht zu vernachlässigenden Entwicklungsbestandteil bei der Programmierung werden. Eine der Hauptaufgaben der Commuity um Freie Software ist das aufspüren, melden und ggf. helfen beim „fixen“ von Bugs.

Community – hier: (meist) virtuelle Gemeinde, die ein bestimmtes Interesse teilt (Vgl. Kapitel 5.3)

> **Compiler** – Ein Programm, welches den in einer höheren Sprache verfaßten Quellcode eines Programms in die Maschinensprache des jeweiligen Prozessors übersetzt. Das Ergebnis ist ein →Binary.

Contributor – siehe →Kontributor

Copyleft – „Copyleft is a general method for making a program free software and requiring all modified and extended versions of the program to be free software as well.“ (Definition der Free Software Foundation; http://www.gnu.org/copyleft/copyleft.html)

Debian – „Das Debian-Projekt ist eine Gemeinschaft von Individuen, die in Gemeinschaftsarbeit ein freies →Betriebssystem entwickeln. Dieses →Betriebssystem, das wir entwickelt haben, wird Debian GNU/Linux genannt, oder einfach nur Debian.“ (siehe: http://www.debian.org/intro/about)

> **Debugging** – Tätigkeit des Suchen Beheben von Programmfehlern; siehe auch →Bug

Distribution – Der Terminus „Distribution“ geht auf die „→Berkeley Software Distribution“ (siehe auch 2.1.2) zurück und wird noch heute als Bezeichnung für die Sammlung, Anpassung und den Vertrieb von Freier Software verwendet; so spricht man zum Beispiel von der SUSE-Distribution oder der →Debian GNU/Linux-Distribution.

Email-Client – Software zum Empfangen, Verfassen und Versenden von Email. Kommuniziert über bestimmte Protokolle wie POP3, IMAP und SMTP mit dem entsprechenden Mail-Server.

Forum – hier: Meist öffentlich Plattform, welche – in machen Fällen mit Zugangsbeschränkungen – über das Internet erreichbar ist und wo Interaktion und Kommunikation stattfinden kann. Sehr „schwammiger“ Begriff, oft auch als „Webforum“ oder „Diskussionsforum“ bezeichnet.

Free Software Foundation (FSF) – Eine Non-Profit-Organisation, welche 1985 von Richard Stallman gegründet wurde zur Unterstützung von Freier Software im Allgemeinen und zur Unterstützung des →GNU-Projektes im Speziellen.

Freeware – Software, welche frei im Sinne von gratis von ihrem Autor vertrieben wird, meist aber ohne →Quellcode.

FSF – siehe →Free Software Foundation

ftp – File Transfer Protocol; Protokoll für die Übertragung von Dateien über das Internet. Siehe auch →Protokoll

GNU General Public License (GPL) – Von der →FSF, insbesondere Richard Stallman und Eben Moglen, erdachte Lizenz, die zum Schutz des →GNU-Projekts geschaffen wurde und heute die am weitesten verbreitete Freie Software-Lizenz ist.

GNU-Projekt – The GNU Project was launched in 1984 to develop a complete UNIX style operating system which is free software: the GNU system. [...] Variants of the GNU operating system, which use the kernel Linux, are now widely used; though these systems are often referred to as “Linux,” they are more accurately called GNU/Linux systems.“ (siehe: http://www.gnu.org/); Vgl.: Kapitel 2.1.3.

GPL – siehe →GNU General Public License

Hacken/Hacker – „The conventional mass media-definition of hackers is usually only one of renegade intellectual bandits who either break into computing system, spread computer viruses, or pirate software: a group the free software developers calls „crackers.“ We, following a long tradition in the free and open source software communities use the term „hacking“ to describe a social practice and individual activity that fuses technological innovation with artistic creation that is guided by a series of ethical principles such as information freedom and privacy.“ (Coleman/Hill 2004, S. 273).
„A person who delights in having an intimate understanding of the internal workings of a system, computers and computer networks in particular.“ (The Jargon File, http://www.catb.org/~esr/jargon/html/H/hacker.html)

Hacker-Ethik/Hacker-Kultur – To be a hacker meant more than just writing programs [...]. It meant writing the best possible programs. It meant sitting at a terminal for 36 hours straight if that's what it took to write the best possible programs. Most importantly, it meant having access to the best possible machines and the most useful information at all times. Hackers spoke openly about changing the world through software, and Stallman learned the instinctual hacker disdain for any obstacle that prevented a hacker from fulfilling this noble cause. Chief among these obstacles were poor software, academic bureaucracy, and selfish behavior.“ (Williams 2002, S. 39)

http – Hyper Text Transfer Protocol; Protokoll für den Austausch von Daten über das Internet, hauptsächlich zur Übertragung von Web-Seiten genutzt. Wurde 1989 am CERN in Genf von Tim Berners-Lee erfunden. Siehe auch →Protokoll.

ICQ – Kommunikationsform, die auf dem gleichnamigen Internet-Protokoll aufbaut und eine Kommunikation über das Internet in Echtzeit mittels Textnachrichten bereitstellt. Der Name ist angelehnt an „I seek you“ [dt.: „Ich suche dich“].

imap – Internet Message Access Protocol. Wird genutzt, um auf Emails auf einem entfernten Rechner zuzugreifen. Siehe auch →Protokoll.

IRC – Internet Relay Chat. Form der (Gruppen-) Kommunikation über das Internet in Echtzeit, basierend auf Text-Nachrichten.

> **Kernel** – Die Software, die die Hardware eines Computers verwaltet und die Fähigkeiten der Hardware →Anwendungsprogrammen zur Verfügung stellt. Bezeichnet den Kern eines →Betriebssystems.

kompilieren – siehe →Compiler

Kontributor – Mitglieder einer Online-Community, welche in irgendeiner Form zu dem entsprechenden Projekt beiträgt. Sie können verschiedenste Funktionen einnehmen, beispielsweise als Übersetzer, als Programmierer oder als Verantwortlicher für den Außenauftritt eines Projektes.

Linux – Bezeichnet den →Kernel des GNU/Linux-Betriebssystems; wird aber oftmals – fälschlicherweise - synonym zur Bezeichnung des →Betriebssystems als Ganzem benutzt. Linux gilt als das erste Freie Software-Projekt, welches ein breite Beachtung auch in nicht-Computer-Medien hervorrief (Vgl. Kapitel 2.1.5)

Mailing-Listen – Von einem Individuum oder einer Organisation betreute Sammlung von Email-Adressen, deren Inhaber jeweils alle Nachrichten (hier: Emails) bekommen, unabhängig von der tatsächlichen Empfängeradresse. So können sehr viele Menschen an einer halb-öffentlichen – als Bedingung gilt die „Abonnierung“ der entsprechenden Mailing-Liste – Diskussion teilhaben. Die Linux-Kernel-Mailing-Liste (http://lkml.org/) beispielsweise ist das hauptsächliche Austauschmedium für die Kernel-Entwicklung.

Maintainer – Bezeichnung für den „Betreuer“ eines Projektes. Die Person, die die Pflege, die Koordination und die grundsätzlichen Entscheidungen bezüglich des „eigenen“ Projektes übernimmt. Es kann durchaus mehrere gleichberechtigte Maintainer eines Projektes geben.

Minix – Unix-ähnliches →Betriebssystem von Andrew Tanenbaum. Entwickelt als Lehr-System für die Universität Amsterdam.

Massachusetts Institute for Technology – Forschungsabteilung der Universität in Cambridge, (Massachusetts USA) mit dem Schwerpunkt Technologie. Siehe http://web.mit.edu

MIT – siehe →Massachusetts Institute for Technology

Multics – „Multics“ (Multiplexed Information and Computing Service), auch der Name UNIX geht indirekt darauf zurück: Nach der Portierung wurde das System spöttisch „→Unices“ („Emasculated Multics is Unics“; „Entmanntes Multic ist Unics“) genannt, da es anfangs

lediglich zwei Benutzer verwalten konnte. Zur Geschichte von Multics: http://www.mit.edu:8001/afs/net/user/srz/www/multics-history.html

Newsgroup – System zum Austausch von Nachrichten über Rechnernetze. Vorzustellen als „schwarze Bretter“ oder Diskussionsforen, die die Nachrichten aufnehmen und für andere Bereitstellen.

Pager – Programm, welches (Text-) Dateien zeilenweise auf einem Drucker oder dem Bildschirm ausgibt. Ein Pager besitzt normalerweise keine Funktionen zum Editieren von Dateien (vgl.: →Texteditor).

Plattform – Bezeichnung für Kombinationen von einer spezifischen Computer-Architektur und einem →Betriebssystem; z.B. Microsoft Windows, UNIX/Linux, DOS, MacOS.

pop – Post Office Protocol. Protokoll, um Emails von einem entfernten Rechner abzurufen und lokal zu speichern. Siehe auch →Protokoll

portabel – siehe →Portabilität

Portabilität – Grad an Plattformunabhänigkeit eines Computer-Programms

POSIX – Portable Operating System Interface for UNIX. Ein Standard für die Schnittstelle zwischen Applikation und dem UNIX-Betriebssystem.

Protokoll – „Sprache“ zwischen Computern. Bezeichnet eine Konvention oder einen Standard, der die Modalitäten der „Sprache“ regelt. Auch dem Endbenutzer bekannte Protokolle sind beispielsweise http, ftp, smtp, pop oder imap

> **Quellcode** – Text eines Programmes in einer höheren Programmiersprache. Um ein solches Program ausführen zu können, muß es zuerst mit einem →Compiler in eine Maschinen-lesbare Sprache übersetzt werden.

Rechner – Synonym: Computer; Überbegriff für unterschiedliche Hardwareplattformen. Gängige Beispiele aus dem Heimbereich sind er Personal Computer, oder Power Macintosh von Apple Computer.

RFC – siehe →Request for Comments

Request for Comments (RFC) – Sammlung von Dokumenten die offene Technologien und Standards des Internet beschreiben. Entstehen in einem offenen Prozeß der Diskussion und sind nicht bindend. Durch ihre technische Überlegenheit hin und ihre sehr weite Verbreitung sind sie zu De-Facto-Standards in bestimmten Bereichen geworden (Vgl.: Protokolle)

Schnittstelle – hier: Der Punkt, wo zwei unterschiedliche Entitäten aufeinandertreffen. Dies kann eine Hardwareschnittstelle wie z.B. USB sein, eine Softwareschnittstelle für die Kommunikation von Software-Komponenten untereinander oder auch Benutzerschnittstellen als der Punkt, an dem der Mensch direkt mit der Maschine interagiert.

Smartphone – Vereint den Leistungsumfang eines Personal Digital Assistant (PDA) mit einem Mobiltelefon.

smtp – Simple Mail Transfer Protocol. Das Standardprotokoll für den Versand von Emails über das Internet. Siehe auch →Protokoll

System-Plattform – siehe →Plattform

TCP/IP – Eine →Protokoll-Familie die zur Kommunikation zwischen Rechnernetzen benutzt wird. Sie steht für zwei Protokolle: Das Transmission Control Protocol und das Internet Protocol. Entstanden in der Universität von Berkeley, Kalifornien, ist es noch heute das meistbenutzte Protokoll um ein Netzwerk aufzubauen und jedes moderne →Betriebssystem „spricht und versteht“ TCP/IP.

Terminalemulator – Software, welche die Ein-/Ausgabe bei der Nutzung eines entfernten Rechners übernimmt.

Texteditor – Programm zum editieren von Textdateien. Wichtiges Werkzeug für Programmierer und UNIX-Administratoren.

Tool – Beispielsweise der in Kapitel 2.1.4 erwähnte Editor Emacs – als „Werkzeuge“ eines Programmierers. Generelle Bezeichnung für Software, die Computernutzern bei der Bewältigung ihrer Arbeit hilft.

Turing-Maschine – Von dem Mathematiker Alan Turing (1912-1954) erdachte Machine, welche jedes algorithmenbasierte Problem zu lösen im Stande ist. Aufgrund dieser „Universalität“ wird sie auch Universalmaschine genannt und gilt als Prädezessor des heutigen Computers.

Unics – →Betriebssystem, Vorläufer von UNIX

Universalmaschine – siehe: →Turing-Maschine

UNIX – Portables, multi-user und multi-tasking →Betriebssystem, ursprünglich in den AT&T Bell Labs entwickelt, siehe Multics, siehe Linux, vgl. Kapitel 2.1.1.

UUCP (UNIX to UNIX copy) – Überholtes →Protokoll zur Übertragung von Dateien zu entfernten Computern, insbesondere UNIX-Rechnern. Sehr frühes (1986) Protokoll zum Austausch elektronischer Post, ähnlich dem, was wir heute als „Email“ bezeichen.

Voice-over-IP (VoIP) - Wird auch IP-Telefonie genannt und beschreibt eine Technik, mit der über ein Computer-Netzwerk wie dem Internet eine Sprachverbindung hergestellt werden kann.

WLAN – Wireless Local Area Network, drahtloses Netzwerk über das höhere →Protokolle wie http oder ssh genutzt werden können.

XFree86 – Eine freie Implementierung des X-Fenster-Systems, der am meist-verbreiteten graphischen Benutzerschnittstelle bei UNIX-Systemen.

Anhang II - Zeitleiste

1950 bis 1969

1953	•	Geburt Richard Stallman's in New York.
1964	•	Das →MIT, General Electric und AT&T entwickeln das →Betriebssystem „→Multics“.
1969	•	Unix wird von Ken Thompson und Dennis Ritchie bei AT&T.
	•	Aufgrund eines kartellrechtlichen Verfahrens wird es IBM untersagt, seine Hardware mit entsprechender Software vorausgestattet zu vertreiben. Dies wird teilweise als Beginn der Geschichte von kommerzieller Software gesehen.
	•	Linus Benedict Torvalds wird in Helsinki geboren.

1970 bis 1979

1970	•	Stallman besucht die Harvard University.
	•	Das →Betriebssystem UNIX wird das erste Mal öffentlich so benannt und wird produktive in den Bell Laboratories eingesetzt.
1971	•	UNIX wird Standard-→Betriebssystem bei AT&T.
	•	Stallman beginnt seine →Hacker-Karriere am →AI-Lab (→MIT).
1973	•	UNIX wird in der Programmiersprache C neu geschrieben und an unterschiedliche Unternehmen und Institutionen lizenziert.
1974	•	Erstes UNIX an der Universität Berkeley in Betrieb.
1975	•	Bill Gates und Paul Allen entwickeln und vertreiben ihr erstes Produkt, einen BASIC-Dialekt für den Altrair 8800-Computer.
	•	Gründung der Microsoft Coorp.

	• Stallman's Emacs wird mit der Bitte veröffentlicht, daß Veränderungen an der Software an Stallman zurückgeschickt werden sollen.
1976	• Bill Gates: Fellow Letter to Hobbyists (Gates 1976).
	• Erste →Berkeley Software Distribution.
	• Gründung von Symbolics Inc. als Spinoff des Artificial Intelligence Laboratory des →MIT sowie Abwerbung durch Symbolics von →AI-Lab-Angestellten.
1980 bis 1989	
1981	• →GNU General Public License Version 1.
	• Microsoft MS-DOS wird ausgeliefert.
1982	• William N. Joy gründet zusammen mit Vinod Khosla, Scott McNealy und Andy Bechtolsheim Sun Microsystems.
1983	• →BSD 4.1 wird durch die Implementation des →TCP/IP-Protokolls netzwerkfähig. TCP/IP wird u.a. auch von Microsoft mit ihrem →Betriebssystem Windows ausgeliefert.
	• Richard Stallman kündigt das →GNU-Projekt an.
1984	• AT&T wird zerschlagen und damit eine kommerzielle Nutzung der UNIX-Quellen ermöglicht.
	• Stallman hört bei →MIT auf, und widmet sich voll dem →GNU-Projekt.
1985	• Stallman setzt das GNU-Manifesto auf.
	• Gründung der Free Software Foundation zur Unterstützung von freien Softrware-Projekten.

1987	• Larry Wall stellt die Version 1.00 der Programmiersprache Perl fertig.
	• Andrew Tanenbaum entwickelt das Lehr-→Betriebssystem Minix.
1988	• Linux beginnt das Studium der Computerwissenschaften an der Universität von Helsinki.
	• Microsoft Windows NT (New Technology) wird eingeführt.
1989	• Erste freie →Berkeley Software Distribution „Networking Release I“.
1990 bis 1999	
1990	• Richard Stallman wird für seine Arbeit am →GNU-Projekt zum „Fellow“ der McArthur-Stiftung ernannt und erhält dafür 230.000 US-Dollar über fünf Jahre.
1991	• Da die GNU →GPL in einigen Bereichen als zu restriktiv empfunden wurde, wird sie modifiziert und als GNU GPL Version 2 veröffentlicht.
	• Microsoft Windows 3.0 wird eingeführt, ebenso Microsoft Visual Basic.
	• Erste öffentliche Linux-Version 0.01.
	• Tim Berners-Lee erfindet das World Wide Web am CERN in Genf.
	• Sun Microsystems entwickelt die Programmiersprache Java.
1992	• Erstes vollständiges und freies →Betriebssystem für den 386er PC: 368/BSD.
	• Erste öffentliche Ankündigung des Samba-Projekts (damals noch unter dem Namen „nbserver“).

	• Microsoft Windows 3.1 wird ausgeliefert, erstmals mit graphischer Benutzerschnittstelle.
	• Aufgrund des zunehmenden Interesses an Linux wird eine eigene →Newsgroup comp.os.linux eingerichtet.
	• Für die Kombination der freien GNU-Programme mit dem freien Linux-Kernel etabliert sich der Begriff GNU/Linux.
	• Gründung "Gesellschaft für Software und Systementwicklung mbH" (SuSE) als größter europäischer Linux-Distributor in Nürnberg.
1993	• CERN gibt die Technologie des World Wide Web frei.
	• Das →Debian-Projekt, als eine komplett aus Freier Software bestehende →Distribution, wird von Ian Murdock gegründet.
1994	• Gründung der GNU/Linux-Distributionen Caldera und Red Hat.
	• Netscape Navigator Version 1.0.
	• Linux Version 1.0.
1995	• Apache Version 1.0.
	• Das Unternehmen Netscape geht an die Börse und löst den „New Economy-Boom“ aus.
1996	• KDE-Projekt gegründet.
1997	• Eric S. Raymond verfaßt die erste Version von „The Cathedral and the Bazaar“.
	• GNOME-Projekt angekündigt.
1998	• Microsoft Windows NT wird als Unternehmensplattform eingeführt und als Konkurrenz zu dem zu diesem Zeitpunkt sehr zersplitterten UNIX aufgestellt.

	• Der Begriff von „Open Source“ soll die als problematisch empfundene Formulierung von „Free Software“ ablösen und für eine höhere Akzeptanz seitens der Wirtschaft sorgen.
	• Gimp Version 1.0.
	• KDE Version 1.0.
1999	• 1999 werden die →Kontributoren des Linux-Kernels für seine Verdienste um eine demokratische Digitalisierung mit dem Prix Ars Electronica ausgezeichnet. Die Jury begründet ihre Entscheidung damit, daß das →Betriebssystem Linux als erstes Produkt aus dem Cyberspace des Internet Auswirkungen auf die reale Welt hat.
2000 bis 2004	
2001	• Free Software Foundation Europe in Deutschland gegründet.
2002	• Mozilla Version 1.0.

---

1. Auf die Unterscheidung zwischen Anwender und Entwickler wird später (Kapitel) näher eingegangen.↩
2. Um den Begriff „Freie Software“ anhand eines konkreten Beispiels zu verdeutlichen, so entstand diese Arbeit auf einem GNU/Linux-System, als Textverarbeitungssoftware wurde OpenOffice eingesetzt. Dieser Freien Software stehen zum Beispiel das Microsoft Windows-Betriebssystem und das Microsoft Office-Paket mit der Textverarbeitung „Word“ gegenüber.↩
3. Im Falle der Nicht-Verfügbarkeit einer der angegebenen Online-Quellen bitte an mail@tombreit.de wenden. Weiter befinden sich die transkribierten Interviews sowie die vorliegende Arbeit selbst in einem hypertext-fähigen Format unter der URL http://www.tombreit.de/vers2/sites/uni/.↩
4. Vgl.: Zur Auslegung von „frei“ in Freier Software, siehe Kapitel 5.4.1. In der vorliegenden Arbeit wird der Terminus „Freie Software“ aufgrund der speziellen Auslegung von „Freiheit“ als Eigenname verwendet und daher groß geschrieben.↩
5. hier: Das →Betriebssystem GNU/Linux.↩
6. In regelmäßigen Abständen untersucht Netcraft die Marktanteile verschiedener Server-Software, dabei hat seit 1996 der Apache Webserver (http://apache.org) alle Konkurrenten weit überholt. Eine jeweils aktuelle Darstellung des Web-Surveys ist unter http://news.netcraft.com/archives/web_server_survey.html einzusehen.↩
7. Das Jahr 1997 wurde als Referenzzeitpunkt gewählt, da damals eine der grundlegenden Studien zu Freier Software und Open Source Software von Eric Raymond (siehe auch Kapitel 4.1) veröffentlicht wurde. Dazu trat damals – wie auch an den aufgeführten Zahlen zu sehen ist – Freie Software ins Blickfeld einer größeren Öffentlichkeit, was als Folge der Geschäftsmodelle zu sehen ist, die sich langsam etablierten und auch zu spektakulären Börsengängen – beispielsweise bei VA Linux (1999) oder RedHat (1999) – führten.↩
8. Analyse von folgenden regionalen, nationalen und internationalen Zeitungen und Zeitschriften, explizit ausgenommen sind Branchen-spezifische Printmedien (in alphabetischer Reihenfolge):
Bayerische Staatszeitung und Bayerischer Staatsanzeiger, Berliner Zeitung, Chrismon Plus, Der Tagesspiegel, Deutsche Welle, Die Zeit, Financial Times Deutschland, Focus, Frankfurter Rundschau, Handelsblatt, Jetzt, Monitor-Dienst, Neue Zürcher Zeitung, NZZ am Sonntag, NZZ Folio, Spiegel, Stern, Süddeutsche Zeitung, SZ-Extra, SZ-Landkreisausgaben, SZ-Magazin, Tageszeitung Regional, Tageszeitung (Analyse anhand des Medienports des Dokumentations- und Informationszentrums München, 2004).↩
9. Vgl.: Die Halloween-Dokumente; siehe Fußnote 77.↩
10. Die Kampagne ist unter http://www.microsoft.com/windowsserversystem/facts/ einzusehen. Des weiteren werden in wichtigen Medien der Branche Anzeigen geschaltet.↩
11. Der Begriff „Zirkel“ betont in der vorliegenden Arbeit die sehr enge Verbundenheit der einzelnen Gruppenmitglieder miteinander und das Teilen eines außergewöhnlichen Interesses bis hin zu einer Elitisierung derselben. Damit soll er sich bewußt von dem soziologischen Gruppenbegriff absetzen, der idealtypisch entweder als Kleingruppe (Hillmann 1994, S. 311) mit bis zu 25 Mitgliedern in Erscheinung tritt, oder aber als Großgruppe (ebd.), welche allerdings organisatorische Maßnahmen für ihre

Aufrechterhaltung im Sinne von Regulierung und Bürokratisierung voraussetzt. Diese Voraussetzungen sind hier nicht gegeben; es kann viel eher von einer Mischform von Klein- und Großgruppe gesprochen werden, da zum Einen die Mitgliederzahl der entsprechenden Gruppen hier die Zahl von 25 weit übersteigt, zum Anderen es sich nicht – wie oben dargestellt – um Gruppen handelt, welche Organisationen zu ihrer formalen Regelung hervorbringen. Insbesondere das Formale ist den hier mit „Zirkeln“ umschriebenen Gruppen fremd. Für die vorliegende Arbeit wurde der Begriff der Zirkel für den geschichtlichen Teil gewählt, um die eigene, „elitäre“ Sphäre der Freien Software-Entwickler gegen die sehr viel offenere Interpretation des Community-Begriffs (siehe Kapitel 5.3) in der aktuellen Freien Software-Diskussion abzugrenzen.↩

12. Für den in der vorliegenden Arbeit verwendeten Ideologie-Begriff, siehe Kapitel 5.4.↩
13. Das Wort „→Universalmaschine“ geht auf Alan Turing's sog. „Universal Turing Maschine“ zurück. Turing gilt als einer der Begründer der Informatik, da er 1936 mit der →Turingmaschine ein „universelles Rechenmodell“ vorlegte (Vgl. zu Person und Werk Turing's: http://www.turing.org.uk/turing/).↩
14. Vgl.: Kapitel 2.1.3; sowie Williams 2002, S. 42ff und Levy 1994, S. 74.↩
15. Vgl.: Tabellarische Chronik in Anhang II – Zeitleiste.↩
16. Vgl.: Grassmuck 2002, S. 202.↩
17. Vgl.: siehe auch Kapitel 3.2; Levy 1994, S. 39ff, S. 413ff.↩
18. Das „Scheitern“ bezieht sich auf die Zusammenarbeit der Bell Labs (AT&T) und dem →MIT an dem Projekt →Multics. Obwohl es in der Literatur fast durchgängig als ein absolutes Scheitern dargestellt wird, existierte auch Multics noch weiter, bis Ende 2000 die letzte Multics-Maschine vom Netz genommen wurde – nicht mehr als Forschungsprojekt, sondern als kommerzielles Produkt von Honywell. Dieses Gerücht – um das Scheitern – hält sich sogar hartnäckig in technischen Kreisen, wie der falsche Eintrag im Jargon File belegt
(Vgl.: http://www.jargon.net/jargonfile/m/Multics.html).↩
19. Bekannte GNU/Linux-Distributoren: http://www.suse.de/de/index.html; http://www.mandrakesoft.com/; http://www.redhat.com/.↩
20. Zur Abgrenzung und Interpretation der hier verwendeten Begriffe „technisch“ und „ideologisch“, siehe Kapitel 5.4.↩
21. http://www.berkeley.edu/↩
22. Fehlerbehebung wird in der Software-Branche als „→Bug-Fixing“ bezeichnet. Die Bezeichnung „Bug“ [dt.: Käfer] geht auf einen Computerabsturz zurück, der durch einen Käfer verursacht wurde, der in einem Rechner für einen Kurzschluß gesorgt hatte und noch heute – allerdings tot – bewundert werden kann. Ausführlich dargestellt in: http://en.wikipedia.org/wiki/Computer_bug.↩
23. Sun Microsystems besitzt heute ein umfassendes Produktportfolio. Sun Microsystems vertreibt eine eigne →UNIX-Variante namens Solaris und entwickelt die Programmiersprache Java (http://www.sun.com/).↩
24. Sehr gute Biographie Richard Stallman's, insbesondere zu den politischen, sozialen und ökonomischen Verflechtungen der Geschichte von Freier Software mit seiner einzigartigen Persönlichkeit bei Sam Williams (Vgl.: Williams 2002).↩
25. Vgl.: „He [Richard Stallman; Anm. d. Verf.] enjoyed the "high" that came near the end of a 20-hour coding bender“ (Williams 2002, S. 56)↩
26. Vgl.: „[...] brilliant programmer [Richard Stallman; Anm. d. Verf.] with the ability to generate large quantities of relatively bug-free code.“ (Williams 2002, S. 60)↩
27. Vgl.: Kapitel 3.2.1.↩
28. Die Firma „Symbolic Inc.“ ist insofern von Bedeutung, da mir ihr das proprietäre Software-Modell Einzug hielt in Richard Stallman's Sphäre der „Freiheit“ und diese derart einschränkte, daß er als Folgen seinen „Crusade [dt.: Kreuzzug] for Free Software“ (Williams 2002, S. 1) begann und bis heute führt.↩
29. Die „virtuelle Heimat“ des →GNU-Projekts findet sich unter http://www.gnu.org/; ins Deutsche übertragen bedeutet diese rekursive Abkürzung soviel wie: GNU ist nicht UNIX.↩
30. Eben Moglen, u.a. Justiziar der Free Software Foundation und Professor für Recht und Rechtsgeschichte an der Columbia Law School der Columbia University in New York, umreißt die Möglichkeiten Freier Software in einem aktuellen Interview folgendermaßen: „Der Grund dafür, dass es sich nur um eine vorübergehende Situation [das Bestehen des Quasi-Monopols Microsofts in der Informationstechnologie; Anm. des Verf.] handelt, begründet sich meiner Meinung nach in Freier Software, also der Existenz einer kostengünstigeren Art, qualitativ bessere Güter herzustellen.“ (Moglen 2004, S. 1)↩
31. Vgl. zur Bedeutung von Freiheit in nicht genuin Software-bezogenen Bereichen: Kapitel 5.4.1.1.↩
32. Vgl.: Stallman 1985.↩
33. Free Software Foundation (FSF): http://www.fsf.org/; Free Software Foundation Europe (FSFE).↩
34. „Just for Fun“ ist auch der Titel von Linus Torvalds Biographie (Torvalds 2001) und soll eine seiner – wichtigsten – Motivationen widerspiegeln.↩
35. Vgl. dazu den Abschnitt zu „Benevolent Dictator“ (nach Rivlin 2003, S. 1) in Kapitel 5.3.↩
36. Nur →Quellcode, d.h. *.c und *.h-Dateien ausgewertet; nur eine sehr grobe Bestimmungsmethode für die Arbeit die in der Entwicklung steckt. Vgl.: Tuomi 2004 für die Problematik der Methode.↩
37. Für mögliche Kategorisierungen und einer eingehenden Behandlung ausgewählter Motivationen siehe Kapitel 5.↩
38. Das →GNU-Projekt ist die Heimat einer ganzen Reihe von einzelnen Freien Software-Programmen, deren Zusammenspiel, deren rechtliche Absicherung und deren Promotion von dem GNU-Projekt koordiniert wird und deren Zusammenfügung das GNU-System, ein freies →Betriebssystem ergibt.↩
39. Vgl.: Kapitel 5.4.1.1.↩
40. WOS: „Das Wizards of OS ist eine in Berlin stattfindende Konferenz, die der Autor Volker Grassmuck in Kooperation mit der Bundeszentrale für politische Bildung organisiert.“ aus: Wikipedia 2004, http://de.wikipedia.org/wiki/WOS↩
41. Zum Begriff der Feedback-Schleife, siehe Kapitel 5.3 sowie Kapitel 5.1.4.↩
42. Vgl.: Diskussion um Freie Software bei der Migration der Stadt München; in
Erik Möller: „Streit der Kulturen“ (Möller 2001).

Süddeutsche 5/2003: „München setzt auf Linux“.↩

43. Hier meist synonym für „Computer“ verwendet. Soll auf die historischen Wurzeln in der „→Universalmaschine“ verweisen; siehe dazu auch Kapitel 2.↩
44. Vgl. für eine Darstellung von Motivationen aus der Perspektive des Anwenders Fußnote 46.↩
45. Hier verstanden im Sinne des Kampfes von „David gegen Goliath“, der Freien Software-Gemeinden gegen die Software-Industrie. Dabei scheint die Microsoft Coorperation (http://microsoft.com) den größten Anteil an der „Anti“-Stimmung zu produzieren und zu provozieren. Zwischen den unterschiedlichen Lagern dieses „Konflikts“ ist mitunter ein regelrechter Glaubenskrieg um die favorisierte →Plattform zu beobachten. Auch das Selbst-Verständnis der Mitarbeiter und Mitglieder des Freien Software-Lagers steuert durch die Formulierung als eine Bewegung, als eine Community, ihren Anteil an Konfliktpotential bei, da diese wertaufgeladene Selbst-Definition sie zum Einen angreifbar macht, zum Anderen eine klare Aussage zu den Grenzen der Bewegung, der Community beinhaltet und damit zu einem „Fremden“ macht. Der Autor ist diesbezüglich nicht unparteiisch, versucht aber diese „Glaubensfragen“ aus der vorliegenden Arbeit herauszuhalten.↩
46. Vgl.: FLOSS-Bericht: „Use of Open Source Software in Firms and Public Instituions: Evidence from Germany, Sweden and UK“ (Wichmann 2002, S. 19ff):
    höhere Unabhängigkeit von der Preis- und Lizenzierungspolitik großer Softwarehersteller
    Wille, die Open Source-Community zu unterstützen
    besserer Zugang zu Spezialisten für Open Source-Software als für proprietäre Software auf dem Arbeitsmarkt
    höhere Stabilität und effizientere Möglichkeiten der Zugriffsbeschränkung für unautorisierten Zugriff↩
47. Diese „weiteren Bereiche“ stehen für sämtliche potentielle Anwendungsszenarien Freier Software in der Kommunikationstechnologie, der Wissensproduktion und Distribution und im wirtschaftlichen Sektor, um lediglich eine Auswahl zu nennen.↩
48. Diese „Realität“ bezieht sich auf den hohen Grad der Durchdringung von Freier Software in der Umwelt. Die meisten angeforderten Web-Seiten werden von Freier Software – hier: Apache Web-Server (http://apache.org) – ausgeliefert, fast jede versendete Email wird auf ihrem Weg zum Empfänger mit Freier Software – hier: dem →Betriebssystem GNU/Linux und der Mail-Transfer-Software sendmail (http://www.sendmail.org/), exim (http://www.exim.org/) oder postfix (http://www.postfix.org/) in Berührung kommen, um nur einige Beispiele zu nennen.↩
49. Vgl. zum Aspekt proprietärer Email-Protokolle auch Fußnote 53.↩
50. Eventuelle Ähnlichkeiten mit lebenden Personen oder Situationen sind beabsichtigt↩
51. Bei Burkhart ist Kommunikator der komplementäre Begriff zum Rezipient; er bezeichnet die aktive Rolle eines Akteurs im Kommunikationsprozess (Vgl.: Burkhart 1995, S. 161).↩
52. Vgl. zu Aspekten der Kontrolle, Fremdbestimmtheit und Offenheit auch Kapitel 5.4.1 sowie Kapitel 5.4.↩
53. „Proprietäre“ →Protokolle zur Email-Kommunikation wurden von den großen Email-Providern und der Content-Industrie bzw. den Portalbetreibern in der Tat eingeführt, jedoch erreichten diese in ihrer damaligen Form keinen größeren Publikumskreis und wurden daher entweder eingestellt oder auf ein offenes Protokoll migriert. Bsp: Compuserv-Mail (eingeführt 1989), Microsoft Exchange (eingeführt 1995) und Lotus Notes (eingeführt 1989).↩
54. Vgl.: Internet Engineering Task Force (http://www.ietf.org/), Internet Society, hier: Geschäftsbereich „Standards“ (http://www.isoc.org/standards/)↩
55. Auch hier wird der offene Entwicklungsprozeß der Internet-Geschichte deutlich: Die Standards entstanden und entstehen aus einem öffentlich ausgetragenen Diskurs, daher der Name „Request for Comments“ oder kurz „→RFC“. Die Strukturen um die RFC's sind sehr gut dargestellt in der Wikipedia, und vermitteln wiederum einen Einblick in den Gedanken der Community-basierten Entwicklung:
    „How RFCs Are Made (The RFC Process)
    The RFCs are produced in a process that is different from that used in formal standards organizations such as ANSI. They can be floated by technical experts acting on their own initiative and reviewed by the Internet at large. Practically speaking, standards-track RFCs are usually produced by experts participating in working groups which first publish what the IETF calls Internet-Drafts; this facilitates initial rounds of review before documents become RFCs.
    The RFC tradition of pragmatic, experience-driven, after-the-fact standard writing done by individuals or small working groups has important advantages over the more formal, committee-driven process typical of ANSI or ISO. […]
    The RFCs are most remarkable for how well they work - they manage to have neither the ambiguities that are usually rife in informal specifications, nor the committee-perpetrated misfeatures that often haunt formal standards, and they define a network that has grown to truly worldwide proportions.". (Vgl.: http://en.wikipedia.org/wiki/Request_for_Comments)↩
56. Software, um auf den Email-Server zuzugreifen. Bsp.: Microsoft Outlook, Netscape Mail, Eudora, Mozilla Thunderbird, Evolution.↩
57. Für Joel West und Jason Derick des Center for Research on Information and Organizations der Universität Irvine stellt die Etablierung von Freien Standards in einer Netzwerk-Ära den entscheidenden Faktor für eine breite Annahme und Anwendung der Produkte und Prinzipien des „Open Source Movements“ (West/Derick 2001, S. 4f) dar.↩
58. Zur Darstellung der ideologischen Motivation und der dazugehörigen Begriffsklärung sei auf Kapitel 5.4 verwiesen.↩
59. http://slashdot.org↩
60. Breit angelegte Studie zur Beseitigung des Mangels an fundierten Informationen über Freie und Open Source Software. Von der Europäischen Kommission in Auftrag gegeben und durch das International Institute of Infonomics der Universität Maastricht sowie Berlecon Research im Zeitraum von Mitte 2001 bis Ende 2002 durchgeführt. Der Bericht „Free/Libre and Open Source Software: Survey and Study.“ liegt nicht als ein Dokument vor, vielmehr werden die einzelnen, weitgehend unabhängigen Teile separat publiziert. Für eine Übersicht, siehe: http://floss.infonomics.nl/report/.↩
61. Obwohl aus Gründen der Lesbarkeit im Text die männliche Form gewählt wurde, beziehen sich die Angaben auf Angehörige beider Geschlechter. Hiermit soll in keinster Weise eine geschlechterspezifische Differenzierung vorgenommen werden.↩
62. Vgl.: „Why are There so Few Female Computer Scientists?“ (Spertus 1991). Sehr frühe Arbeit zu dieser Problematik, entstanden am →MIT über die Geschlechterverteilung im MIT.↩

63. FLOSS: Free/Libre and Open Source Software: Survey and Study (Wichmann/Ghosh/Glott 2002)↩
64. Vgl.: „Debian Social Contract“ sowie „The Debian Free Software Guidelines“, hier insbesondere Punkt fünf und sechs. Beide Dokumente unter http://www.debian.org/social_contract↩
65. http://women.alioth.debian.org/↩
66. http://www.linuxchix.org/↩
67. Ein weiteres Dokument existiert seit 2002 als „HowTo“, also in Form der klassischen UNIX-Dokumentation für spezielle Bereiche und trägt den treffenden Titel: „Howto encourage Women in Linux“ (Henson 2002) und wird vom offiziellen „Linux Documentation Project“ gehostet.↩
68. Bettina Unger geht in ihrer Diplomarbeit beispielsweise davon aus, daß EU-weit durchschnittlich 30 % der Arbeitsstellen im IT-Sektors von Frauen belegt werden. Die Bedeutung dieses Wertes ist allerdings noch weiteren Einschränkungen unterworfen: So wird u.a. lediglich ein 17 %-iger Anteil von Frauen berichtet, die eine Arbeit im IT-Sektor in Bereichen mit hohen Qualifikationsbarrieren gefunden haben. (Unger 2003, S. 30f)
    Eine Studie der Akademie für Technikfolgenabschätzung in Baden-Württemberg berichtet ebenfalls von einem 30 %-igen Frauenanteil an sozialversicherungspflichtigen IT-Beschäftigten in Baden Württemberg. Der Anteil von Frauen in technischen und informatischen Studiengängen beträgt je nach Fachrichtung ca. 10 %, wobei die Mathematik mit einem Frauen-Anteil von über 30 % heraussticht. (Vgl.: Huber et al. 2001)
    Das Senatsamt für die Gleichstellung (Hamburg) hat einen umfangreichen Bericht zu diesem Thema publiziert. An dieser Stelle interessant sind die angeführten Zahlen zu Frauen in IT- oder Multimedia-Berufen. Auch hier ein durchschnittlicher Frauenanteil von ca. 30 % angenommen werden, wobei die Werte zu einzelnen Berufsgruppen sehr streuen. Für GraphikerInnen wird beispielsweise ein Wert von 50 % berichtet, bei ElektroingenieurInnen dagegen nur von einem Frauenanteil von knapp 5 % gesprochen. (Vgl.: Möller 2002, S. 35f)↩
69. http://www.oneeyedcrow.net/resume-geek.html↩
70. Vgl.: Die Rolle der Universität zu Berkeley (Kapitel 2.1.2) und das →MIT und die Arbeit Richard Stallman's (siehe auch Kapitel 2.1.3); sowie die „studentische“ Arbeit Linus Torvalds am dem frühen Linux-Kernel während seines Informatik-Studiums an der Universität Helsinki (siehe auch Kapitel 2.1.5)↩
71. Speziell die Schriften von Eric Raymond (in: Raymond 2001a); eine sehr gute Zusammenfassung von Aufsätzen wichtiger Freier Software-Pioniere bei DiBona et al. (1999).↩
72. Eric Raymond hat als erster das Entwicklungsmodell von Freier Software einer genaueren Analyse unterzogen. In seinem Werk „The Cathedral and the Bazaar“ (Vgl.: Raymond 2001a) führt er die Unterscheidung nach der Art der Produktion von Freier Software nach dem „Kathedralen-Modell“ oder dem „Basar-Modell“ ein. Demnach ist das Kathedralen-Modell gekennzeichnet durch die relative Geschlossenheit der Entwicklungsmannschaft sowie die straffen Strukturen, das Basar-Modell dagegen als vollkommen offenes Modell, lediglich mit einer sehr losen Führung ausgestattet. Er führt das →GNU-Projekt und proprietäre Software als Beispiel für das Kathedralen-Modell, sowie den Linux-Kernel als Beispiel für das Basar-Modell an. Nikolai Bezroukov unterzieht vor allem das Modell des Basars einer kritischen Analyse. Er schließt mit der Überzeugung, „[...] that an academic model of OSS [Open Source Software; Anm. d. Verf.] explains the phenomenon much better [als das Basar-Modell; Anm. d. Verf.].“ (Bezroukov 1999, S. 28).↩
73. Vgl.:
    Im deutschen Sprachraum ist das Werk von Volker Grassmuck „Freie Software. Zwischen Privat- und Gemeineigentum.“ hervorzuheben. Es stellt klar und gut strukturiert die Freie Software und seine historischen wie aktuellen Umgebungsvariablen dar. Dabei wird auf Fragen des Eigentums ebenso eingegangen wie auf wirtschaftliches Potential oder gesellschaftliche Relevanz auch außerhalb der Ersten Welt. Den Abhandlungen über Freie Software ist ein ausführlicher Teil über die rechtliche Ordnung von Wissen vorangestellt. (Grassmuck 2002)
    Das Kompendium über die historische Entwicklung von Freier Software, Open Source Software und Linux im Speziellen liegt mit Glyn Moody's „Die Software Rebellen.“ vor. (Moody 2001)
    Die Biographie von Linus Torvalds, dem „Erfinder“ von Linux. (Torvalds/Diamond 2001)↩
74. Wichtige Arbeiten zur Motivation in Freier Software (Auszug):
    Eric Raymond 1998: „The Cathedral and the Bazaar.“
    Die Fallstudie zum Entwicklungsmodell von Open Source Software; kann als Klassiker bezeichnet werden.
    DiBona et al: 1999: „Open Sources: Voices from the Open Source Revolution.“ (Sammelwerk mit Aufsätzen der Größen von Freier Software und Open Source Software)
    Lerner and Tirol 2000: „The simple Economics of Open Source.“
    Ökonomische Analyse der Motivation Freier Software-Entwickler
    Lakhani, von Hippel 2003: „How open source software works: „free“ user-to-user assistance.“
    Studie zur Motivation von „banalen“ Tätigkeiten in Freien Software-Projekten
    Sandeep Krishnamurthy 2002: „Cave or Community? An Empirical Examination of 100 Mature Open Source Projects"
    Empirische Studie zur Theorie des Basar-Modells (Vgl. Raymond 1998) und Widerlegung dieses Modells.
    Steven Weber 2004: „The Success of Open Source.“
    Sam Williams 2002: „Free as in Freedom. Richard Stallman's Crusade for Free Software.“↩
75. Die folgenden Aussagen des Microsoft-Pressesprechers sind somit nicht wissenschaftlich belegbar; sollen aber nicht gänzlich vorenthalten werden:
    Die Freie Software-Bewegung wurde – trotz ihrer auch für den Konzern Microsoft bekannten Gefahren (s.u.) – nicht ernst genommen. Auch die Frage nach möglichen Motivationen wurde heruntergebrochen auf ausschließlich Spaß-basierte Aspekte wie das gemeinsame →Hacken und der Wunsch es „den Großen einmal zu zeigen“. Sehr aufschlußreich waren zwei Bemerkungen zu möglichen ideologischen Motivationen. So äußerte sich der Interviewpartner definitiv ablehnend gegenüber der Aussage, daß man von der Arbeit anderer profitiere und eigenes Engagement in diesen Pool wieder zurückgeben möchte (Vgl.: Kapitel 5.2). Ebenso wurden nicht-monetäre Anreize als Motivation mit den Worten: „Warum sonst [durch den Reputationszugewinn und damit erhöhten Karrierechancen; Anm. d. Verf.] sollte man so etwas tun?“ abgelehnt.

Bezüglich der Bedrohung durch das Open Source-Entwicklungsmodell:
Bei den sogenannten Halloween-Dokumenten handelt es sich um vertrauliche Dokumente des Unternehmens Microsoft (http://www.microsoft.com), welche Personen außerhalb dieses Unternehmens in die Hände gespielt wurden. Die ersten Halloween-Dokumente stammen aus dem Jahre 1998 und behandeln die Bedrohung einiger Microsoft-Produkte durch Freie- und Open Source-Software. Die Open Source Initiative zählt bislang 11 Halloween-Dokumente, das letzte aus dem Jahre 2004, deren Inhalt jeweils konträr zu der offiziellen Presse des Unternehmens Microsoft steht. Diese Dokumente werden gesammelt und in dokumentierter Form unter http://www.opensource.org/halloween/ der Öffentlichkeit zugänglich gemacht.↩

76. Vgl. die Ausführungen zum Motivations- und Engagementsbegriffs unter Kapitel 5.4.3.↩
77. Maslow spricht in diesem Zusammenhang von Selbstaktualisierung (Maslow 1978, S. 154).↩
78. Vgl.: Das Flow-Konzept: Csikszentmihalyi (1990) sowie die konkrete Anwendung auf Open Source-Software: Projekt FASD der Universität Zürich, Lehrstuhl für Unternehmensführung- und Politik, http://www.ifbf.unizh.ch/webFuehrung/blprojects/FASD/.↩
79. Der Psychologe Frederick Herzberg zielte mit der Formulierung der „Zwei-Faktoren-Theorie“ auf das Zusammenwirken von in der Tätigkeit selber liegenden Motivatoren (Herzberg 1959, S. 113ff) und in dem Arbeitsumfeld liegenden Hygienefaktoren (ebd.) ab. Auch wenn Herzberg eine psychologische Perspektive einnimmt, kann er soziale Faktoren – insbesondere bei den Hygienefaktoren – nicht ausklammern. Herzberg adressiert die Frage nach einem günstigen Umfeld für effizientes Arbeiten in Gruppen und formuliert dabei Leitsätze, wie sie in der Freien Software-Entwicklung teilweise in Idealausformung auftreten. Die Theorien von Herzberg haben heute – ebenso wie jene von Maslow – einen festen Platz in der Organisationspsychologie und diversen ökonomischen Personal- und Managementtheorien.↩
80. Vgl.: Figure 4-1 in: Inglehart 1990, S. 134.↩
81. Vgl.: Sehr guter Überblick – aus der Perspektive der Freien Software-Community - zu der Verbindung von Freier Software und monetären Aspekten sowie Möglichkeiten der Nutzung und nicht-monetären Kontribution bei Mary Gardiner:
Neben dem direkt finanziellen Engagement steht der Stiftungsgedanke von Ressourcen im Vordergrund. Sehr gut die Zusammenfassungen zu:
„Howto contribute money“
„Howto contribute ressources“; beispielsweise →Bandbreite oder Hardware
„Howto contribute time and skills“; beispielsweise Übersetzungen, Design, Dokumentation, →Debugging
„Howto contribute support“; beispielsweise die Eröffnung eines Online-→Forums oder einer →Mailing-Liste zu einer bestimmten Thematik um Freie Software
„Howto contribute appreciation“; Danksagungen, Lob, generell: positives Feedback an die entsprechenden Entwickler „zurückgeben“ (Gardiner 2003, S. 4-12).↩
82. →Debian kann als eines der größten Freien Software-Projekte angesehen werden, wenn man die Zahl der offiziell Beteiligten zugrunde legt.↩
83. Vgl.: Herzberg (1959) für eine die grundsätzliche Erkenntnis des Zusammenspiels unterschiedlicher Motivations-Faktoren auf psychologischer Ebene; Evers (2001, S. 37-44) für die Hervorhebung der kulturellen Einflüsse auf freiwilliges Engagement.↩
84. Vgl. zu Motivation aus psychologischer Perspektive (in Auswahl):
Helmut Crott: „Soziale Interaktion und Gruppenprozesse.“ Insbesondere die Abschnitte zu Motivation (Crott 1979, S. 75ff) sowie zu kollektiver Leistung (ebd., S. 91ff).
Bernard Weiner: „Theorien der Motivation.“ Hier die Ausführungen zu hedonistischen Modellen (Weiner 1976, S. 108f) und zu Leistungsmotivation in der Gesellschaft, unter Hinzunahme von Weber, Feistinger und Zajonc (ebd., S. 110ff).
Falko Rheinberg: „ Motivation“. Rheinberg behandelt sehr ausführlich verschiedenste Ansätze der Motivationspsychologie und setzt sie oft in einen größeren gesellschaftlichen Kontext; etwa, wenn er Leistungsmotivation u.a. aus der Perspektive der Protestantischen Ethik (Rheinberg 2002, S. 55ff) erklärt und den Umweltbezug (ebd., S. 41f) von Motivation thematisiert. Auch wenn Rheinberg in keiner Weise Freie Software thematisiert, sind seine Analysen fast prädestiniert zur Erklärung des Phänomens Freier Software. Besondere Beachtung verdient, daß hier eine Reihe unabhängiger psychologischer Motivationstheorien zur Erklärung herangezogen werden können. Auch scheint Freie Software als oft „idealtypische“ Ausprägung einer positiven Umsetzung der jeweiligen Motivationstheorien.
Udo Rudolph: „Motivationspsychologie“. Guter, ausführlicher Abriß von Motivationtheorien, dargestellt in der Reihenfolge ihrer historischen Entstehung. Für Freie Software insbesondere wegen der Ausführungen zum reziproken Altruismus (Rudolph 2003, S. 234ff) interessant.↩
85. Die Kategorisierung nach „extrinsischer“ und „intrinsischer“ Motivation findet sich auch in den folgenden Schriften zur Motivation in Freier Software wieder (in Auswahl):
Alfie Kohn: „Studies Find Reward Often No Motivator. Creativity and intrinsic interest diminish if task is done for gain.“ (Kohn 1987)
Karim Lakhani und Robert Wolf: „ Why Hackers Do What They Do: Understanding Motivation Effort in Free/Open Source Software Projects.“ (Lakhani/Wolf 2003)
Richard Ryan und Edward Deci: „Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being.“ (Ryan/Deci 2000)↩
86. Nach Herzberg's Zwei-Faktoren-Theorie gibt es Motivationsfaktoren, die in der Arbeit selber liegen, und deren Erfüllung Zufriedenheit bewirkt, wogegen der andere Pol durch (De-) Motivationsfaktoren bestimmt ist, die durch das Arbeitsumfeld gegeben sind. Herzberg bezeichnet den erstgenannten Faktor als „Motivator“, den letztgenannten als „Hygienefaktor“ (Vgl.: Herzberg 1959). Eine gelungene und wesentlich verständlichere Übertragung dieser beiden Begriffe ins Deutsche unternimmt z.B. Prof. Dr. Stangl von der Universität Linz mit „Kontentfaktoren“ und „Kontextfaktoren“ (Stangl 2004, S. 7).↩
87. Zur Variabilität der Motivationen siehe auch Kapitel 5.4 sowie zusammenfassend Kapitel 6.2.1↩
88. Vgl.: Kapitel 2.1.3; sowie Williams 2002.↩
89. Vgl. Welte 2004, S. 5f.↩

90. Der Vergleich zwischen der Produktion von Freier Software und der von „traditioneller“ Kunst ist nicht perfekt: Kunst wird in der Regel nicht lediglich gegen einen Unkostenbeitrag oder gar gratis vertrieben. Weiter wird in der Kunst der Einblick in das „Handwerk“, die Methoden und Techniken des Künstlers, oftmals kein Einblick gewährt. Auch werden abgeleitete Werke wesentlich „unfreier“ gehandhabt, als dies in Freier Software der Fall, sogar die Intention, ist.↩
91. Vgl.: „A dwarf standing on the shoulders of giants may see farther than the giant himself.“ (Merton 1980, S. 3).↩
92. Verzeichnis der Verweise von wissenschaftlichen Artikeln zu anderen Artikeln und Werken. Aufgrund der Anzahl der Verweise auf bestimmte Werke läßt sich eine „Rangfolge“ der Wertung dieser Werke aufstellen. Der bekannteste und verbreitetste Citation Index ist heute der „Web of Science“ (http://www.isinet.com/products/citation/wos/) des Institutes for Scientific Information der Thomson Cooperation.↩
93. Vgl. dazu die Bemerkungen zu den Changelog- und Credit-Dateien in Kapitel 5.1.4.↩
94. Vgl. auch Barbera 1999, S. 1; Benkler 2002, S. 14ff.↩
95. http://en.wikipedia.org/↩
96. http://slashdot.org/↩
97. http://www.gutenberg.org/↩
98. Altruismus oder altruistisches Verhalten bezieht sich in dieser Arbeit fast durchgängig auf die spezifische Form des reziproken Altruismus. Der reziproke Altruismus versucht egoistische Prädispositionen und altruistisches Verhalten zusammenzubringen, indem er von einer verteilten sowie verzögerten Auszahlungsmatrix ausgeht. Darunter ist zu verstehen, daß derjenige, der sich vermeintlich altruistisch verhält, darauf hofft, daß man sich ihm gegenüber ebenfalls so verhalten wird. Diese Reziprozität ist aber nicht an das Individuum gebunden, für welches man beispielsweise Hilfe geleistet hat, sie definiert vielmehr eine Erwartungshaltung der Gemeinschaft gegenüber, in sich der in diesem Beispiel Hilfeleistende und der Hilfesuchende aufhalten. Der Aspekt der verzögerten Auszahlung bezieht sich auf die zeitliche Dimension: Es muß somit kein direkter zeitlicher Zusammenhang zwischen Leistung und Gegenleistung bestehen, ebenso – wie schon erwähnt wurde – keine direkte Beziehung zwischen dem Hilfeleistenden und dem Hilfesuchenden bestehen muß. (Vgl.: Kapitel 5.2)↩
99. Vgl.: Raymond 2001 und den Begriff der „Noosphere“, der in Raymond's Analyse über die Eigentümerschaft „[...] the space of all possible thoughts“ (Raymond 2001, S. 78) beschreibt. Dieser Entgrenzung auf dem Gebiet der Möglichkeiten der Programmierbarkeit steht die hier behandelte entgrenzte Öffentlichkeit gegenüber, wobei sich der Grad dieser Entgrenzungen jeweils parallel verschiebt.↩
100. Wirtschaftsmagazin brand eins; auch unter: http://brandeins.de/.↩
101. Vgl.: Rossi 2004, S. 3f.↩
102. Die bei den Anwendern liegende Entscheidung, welche Projekte oder welche Entwicklungszweige eines Projekts adoptiert werden, stellt ihre Macht dar (Vgl.: Kapitel 3).↩
103. Die Software des X-Projekts ist für die Darstellung einer graphischen Benutzerschnittstelle verantwortlich und nimmt eine regelrechte Monopolstellung in dem Markt für graphische Fenstersysteme für UNIX-basierte →Betriebssysteme ein. Das „ursprüngliche“ Projekt: http://www.xfree86.org/ und das „geforkte“ Projekt: http://x.org/.↩
104. „All advertising materials mentioning features or use of this software must display the following acknowledgement: This product includes software developed by the University of California, Berkeley and its contributors.“ Entnommen aus der original BSD-Lizenz, siehe:
ftp://ftp.cs.berkeley.edu/pub/4bsd/README.Impt.License.Change.↩
105. Der Lizenztext ist einzusehen unter: http://www.xfree86.org/legal/licenses.html.↩
106. X.org (http://www.x.org/), unter der Schirmherrschaft des Standardisierungsgremiums von Freedesktop.org (http://freedesktop.org/).↩
107. SUSE (http://suse.de) – inzwischen von Novell (http://www.novell.com/d) aufgekauft – und RedHat (http://www.redhat.com/) sind die beiden verbreitetsten GNU/Linux-Distributionen.↩
108. Vgl. zum Hijacking von Open Source-Projekten: Ciffolilli 2004, S. 4ff.↩
109. Vgl.: Fußnote 88.↩
110. Binärversionen von Software entsprechen dem Objektcode des Programms, im Gegensatz zu dem →Quellcode. Binärversionen gelten als nicht les- oder „verstehbar“ für Menschen, sie repräsentieren die Sprache der Computer-→Plattform, auf dem die entsprechende Binärversion ausgeführt werden soll. Das Prinzip Freie Software funktioniert ausschließlich, wenn der Quellcode eines Programmes verfügbar ist, da nur durch das Studium und die Modifizierbarkeit des Quellcodes eine Modifikation des Programms möglich ist. (Vgl. die vier Freiheiten in Freier Software, festgeschrieben in der →GPL in Kapitel 5.4.1.↩
111. Bei dem Software-Archiv freshmeat (http://freshmeat.net/) sind über 68 % der insgesamt 35.433 Software-Projekten unter der →GPL lizenziert. Die nächste Platzierung belegt die LGPL mit einem Anteil von knapp 6 % (Stand: 05.12.2004). 69 % der auf sourceforge (weiteres bedeutendes Portal für Software-Entwicklung; http://sourceforge.net) gehosteten Software ist ebenfalls unter der GPL verfügbar.↩
112. http://www.sitecom.com/↩
113. http://netfilter.org/↩
114. Der Originalwortlaut der richterlichen Entscheidung unter: http://www.ifross.de/ifross_html/eVWelte.pdf↩
115. SCO Group, ehemals Caldera Systems; UNIX-Systemhaus; http://sco.com/scosource/↩
116. Gute Übersicht vom heise-Verlag: „SCO vs. Linux: Die unendliche Geschichte“, http://www.heise.de/ct/aktuell/meldung/44492, siehe die „unendliche“ Link-Liste auf der Seite; guter Einblick in die Evolution von Linux und konkurrierenden Systemen.↩
117. International Business Machines: http://www.ibm.com/↩
118. http://www.novell.com/↩
119. http://www.groklaw.net/↩
120. Für die ausführliche Darstellung sei auf Raymond 2001b, S. 71ff verwiesen.↩

121. Einer der zentralen Begriffe der vorliegenden Arbeit. Reputation verweist auf das Bedürfnis nach Anerkennung und wird unter Kapitel 5.1 detailliert behandelt.↩
122. Anerkennung kann nicht nur für das Schreiben von →Quellcode erworben werden. Auch organisatorische Tätigkeiten oder das Verfassen von Dokumentation wird von der Freien Software-Gemeinde, aber besonders von der jeweiligen Projekt-Community honoriert. Die im Text gemachten Aussagen zu Konkurrenz treffen auf diese Gebiete ebenso uneingeschränkt zu wie bezüglich der Qualität des Quelltextes.↩
123. Eric Raymond verweist darauf, daß beispielsweise Kritik in Freier Software nie Personen- sondern immer Projekt-bezogen ist (Raymond 2001b, S. 110).↩
124. Vgl. zur Selbstlosigkeit die Aussage von Werner Koch, der für die Arbeit an Freier Software ein Beschäftigungsverhältnis verließ, um mehr bzw. nur noch an Freier Software zu arbeiten, und dabei „auch so einen Großteil [seiner; Anm. d. Verf.] Ersparnisse damals aufgebraucht.“ (Koch 2004, S. 5) hat. Leider kann nicht festgestellt werden, ob das hier beschriebene Verhalten eher altruistisch, ideologische (siehe Kapitel 5.4) oder hedonistisch motiviert ist.↩
125. Vgl. zur Rücksichtnahme gegenüber Dritten: „[...] und merkte, daß andere Leute die selben Probleme haben, stellen immer die gleichen Fragen, und dann habe ich mir gedacht, daß es einfach sinnvoll wäre, das alles einmal zusammenzuschreiben.“ (Michlmayr 2004, S. 9); und weiter: „Ja, und eben, wenn ich etwas verbessere, dann profitiere nicht nur ich oder eine Firma davon, sondern eben jeder kann dann potentiell davon profitieren.“ (ebd., S. 12).↩
126. Im Gegensatz zur einer evolutionären Perspektive auf altruistisches Verhalten: Hier wäre Altruismus eher als Verhaltenszug zu verstehen, als „natürliche“ Reaktion auf bestimmte Situationen. Im Speziellen würde altruistisches Verhalten der reproduktiven Fitness dienen, und würde somit weniger einer kognitiven Kontrolle im Sinne eines individuellen Auswahlprozesses zwischen unterschiedlichen Verhaltensalternativen unterliegen.↩
127. Vgl.: Die Ausführungen von Lerner und Tirole zu den Hintergründen dieser bilanztechnischen Darstellung als „net benefit“ (Lerner/Tirole 2000, S. 14) sowie Kapitel 5.1.↩
128. David Lancashire beschäftigt in seiner empirischen Analyse „The Fading Altruism of Open Source Development“ mit konkurrierenden Erklärungsansätzen, speziell der klassischen ökonomischen Theorie und der Schumpeter'schen Annahme, daß die Effizienz einer industriellen Produktionsfunktion ohne die sie umgebenden sozialen Institutionen gemessen werden kann. In der Folge behandelt er die Opportunitätskosten eines Engagements in Freier Software als primären Erklärungsansatz. (Lancashire 2001, S. 2ff)↩
129. Boston Consulting Group; http://bcg.com.↩
130. Open Source Technology Group (OSTG); http://www.ostg.com/.↩
131. Reife in Software-Projekten bezieht sich auf den Status der Software; es lassen sich Einordnungen nach „alpha“ [dt: sehr frühes, experimentelles Stadium], „beta“ [dt: benutzbar, jedoch noch mit bekannten, nicht-behobenen Problemen behaftet] und „mature“ [dt: ausgereift, gilt als stabil] durchführen.↩
132. Vgl.: Lerner/Tirole 2000, S. 14; Prüfer 2004, S. 2ff.↩
133. Vgl. weiter Community (Kapitel 5.3), Ideologie (Kapitel 5.4), sowie ferner Hedonismus, Weiterbildung und Mangel (Kapitel 5.5 bis 5.7).↩
134. Vgl. hierzu die Ausführungen des Soziologen Kieran Healy, der eine Erklärung für empirische Variationen betreffend der Eintrittswahrscheinlichkeit von altruistischem Verhalten aus der soziologisch-organisationstheoretischen Perspektive sucht. Dabei analysiert er die Vorgänge um die Organspende. Seine Untersuchung über die Motivationen dieses Verhaltens – einer Organspende einzuwilligen – an den Tag zu legen, zeigt eine Reihe von Parallelen zu den Beobachtungen des Verhaltens in der Produktion von Freier Software. Healy kommt zu dem Schluß, daß „[i]t is clear, however, that most of the altruism we observe in modern societies is more likely than not to have a strongly institutionalized aspect, with staffed organizations working to produce contexts in which it can happen.“ (Healy 2004, S. 6).↩
135. Vgl. zur Thematik der Absicherung von Freier Software die Auflistung der vier Freiheiten der am weitesten verbreiteten Freien Software-Lizenz, der →GPL in Kapitel 5.4.1 sowie Kapitel 6.3.↩
136. Prof. Dr. Bernd Schauenberg (Universität Freiburg, Personal- und Organisationsökonomie) bei einer Diskussion über Motivation in Open Source-Software Mitte 2004.↩
137. In Anlehnung an den Slogan des Wahlkampfstrategen James Carville für Bill Clinton's US-Präsidentschafts-Wahlkampf 1993: „It's the economy, stupid.“ Man wollte die Aufmerksamkeit der Öffentlichkeit – mit Erfolg – auf die entscheidende Thematik – die Wirtschaft – lenken. Eine ähnlich herausragende Stellung nimmt das Thema „Community“ bei der Frage nach Freier Software ein, denn letztlich ist die Community, die sich um ein Projekt findet, entscheidend für die weitere Existenz und den Erfolg des Projektes. Die Frage nach der Community ist für Freie Software ebenso zentral wie es die Frage nach der Wirtschaft 1993 für den US-amerikanischen Präsidentschaftskandidaten war.↩
138. Die Betonung und Relevanz des Community-Aspekts für das Freie Software-Modell in der vorliegenden Arbeit und weiterer Literatur wird nicht in allen Studien übernommen. Das empirische Material dazu ist oft widersprüchlich. So kommt Sandeep Krishnamurthy zu dem Schluß, daß das „[...] community model is a poor fit for the actual production of the [Free/Open Source; Anm. d. Verf.] software.“ (Krishnamurthy 2002, S. 7) Dem gegenüber stellt Hertel et al. fest, daß – zumindest im Bereich der Linux-Kernel-Entwicklung – 59 % der Beteiligten in Teams mit durchschnittlich 12 Mitgliedern arbeiten. (Hertel/Niedner/Herrmann 2001, S. 19f)↩
139. Vgl.: Der Nerd; Kapitel 3.2.2.↩
140. aus: http://en.wikipedia.org/wiki/Community.↩
141. Raymond führt detailliert auf, auf welche Weise in Freier Software Anerkennung zugestanden wird:
    Die Software hat nur einen Wert – für den Empfänger dieses Geschenks – wenn sie den Erwartungen des Beschenkten gerecht wird.
    Software, welche den Bereich der Freien Software erweitert, verdient mehr Anerkennung als Software, die schon vorhandene Funktionalität neu implementiert.
    Software, welche von bedeutenden →Distributionen aufgenommen wird, ist „mehr Wert“ als Software, die dies nicht erreicht.

„Utilization is the sincerest form of flattery [dt.: Kompliment]“
Mühevolle, „undankbare“ Arbeit wie das Schreiben von Dokumentation oder das →Debuggen verdient und bekommt mehr Anerkennung als das „cherrypicking the fun and easy hacks“.
Vgl.: Raymond 2001b, S. 94ff; auch mit Beispielen für jeden der aufgeführten Punkte.↩

142. Jürgen Habermas anläßlich der Verleihung des Kyoto-Preises am 10 November 2004 in Kyoto, Japan.↩
143. Vgl.: Kapitel 5.1, 5.5, 5.6.↩
144. Überspitzt findet sich diese Leistungsorientierung, diese Selbst-Definition aufgrund des eignen Schaffens auch in Max Weber's Ausführungen zur Protestantischen Ethik: „Er 'hat nichts' von seinem Reichtum für seine Person, – außer: der irrationalen Empfindung 'guter Berufserfüllung'.“ (Weber 1969, S. 60) Dieser Gedanke der „guten Berufserfüllung“ ist in Freier Software am ehesten mit einem den Werten und Wünschen der Community entsprechenden Verhaltens interpretiert.↩
145. Siehe auch Kapitel 5 und 5.3.12.↩
146. Den Begriff der Gruppenautorität verwendet Popitz, um auszudrücken, daß eine Gruppe als Ganzes als „Hüter der Zugehörigkeit“ (Popitz 1987, S. 638) auftritt. Demnach kann jedes Mitglied einer solchen Gruppe eine autoritäre Funktion gegenüber einem weiteren Gruppenmitglied oder gegenüber Dritten wahrnehmen und über Zugehörigkeit oder Abstoßung/Ausstoßung entscheiden.↩
147. Die fünf Formen sozialer Subjektivität nach Popitz (Vgl.: Popitz 1987, S. 637ff):
Bedürfnis nach Zugehörigkeit
Anerkennung in einer zugeschriebenen Rolle
Anerkennung in einer erworbenen Rolle
Anerkennung in einer öffentlichen Rolle
Anerkennung der sozialen Individualität.↩
148. Der Gruppenbegriff in diesem Abschnitt bezieht sich auf jeweils einzelne Gruppen um Freie Software-Projekte und nicht auf die Gesamtheit der Mitglieder der Freien Software-Bewegung – auch wenn sich einige Erkenntnisse auf diese Gesamtheit übertragen ließen. Insgesamt stellt sich diese Gesamtheit als ein Pool von zu heterogenen Akteuren dar, als daß sich generalisierbare Aussagen ableiten ließen.↩
149. Vgl.: Zeitleiste unter Anhang II – Zeitleiste.↩
150. Zu den einzelnen Bedürfnissen siehe Kapitel 5.1, 5.3, 5.5.↩
151. Zum Begriff der Ideologie in der vorliegenden Arbeit siehe Kapitel 5.4.↩
152. In Anlehnung an Leon Feistinger's Theorie der kognitiven Dissonanz; dargestellt in Weiner 1976, S. 151ff.↩
153. Der Begriff der Balance ist hier gewählt, da sich die Theorie der kognitiven Dissonanz als kognitive Balancetheorie – zurückgehend auf Fritz Heider (in: Weiner 1976, S. 148ff) – in eine Reihe weiterer Balancetheorien einreiht.↩
154. Insbesondere der ideologisch motivierte Flügel von Freier Software, repräsentiert durch die →Free Software Foundation, betrachtet sich als Teil einer sozialen Bewegung, welche weiter reicht als lediglich ein Modell zur Produktion von Software bereitzustellen. Diese Betonung der „Bewegung“ in Abgrenzung zu einem Verständnis als rein ökonomische Produktionsfunktion ist aus Perspektive der Free Software Foundation anschaulich dargestellt in der Debatte um die Begrifflichkeiten von „Free Software“ versus „Open Source Software“; (vgl.: FSF 2001).↩
155. Wolf Lotter in brand eins (Lotter 2001, S. 32).↩
156. Korrekt wäre die Verwendung des Plurals von Lizenz, da sich im Laufe der Zeit eine Reihe weitere freie Software-Lizenzen – man beachte das hier klein geschriebene „freie“ – herausgebildet haben, jeweils mit leichten Verschiebungen in der Gewichtung der einzelnen Freiheiten. Da die →GPL als Lizenz der Freien Software im Sinne der →Free Software Foundation in dieser Arbeit zugrunde gelegt wird und diese Lizenz auch die weiteste Verbreitung genießt, wird hier der Singular von Lizenz benutzt.↩
157. Aus der →GNU General Public License (GPL); in: FSF 2004, S. 1.↩
158. Von Robert K. Merton geprägter Begriff zur Abgrenzung von klassischer Güterwirtschaft und der Wirtschaft von „Ideen“. Merton nutzte den Begriff um die Unterschiede zwischen besagter Güterwirtschaft und dem Wissen als Resultat wissenschaftlichen Arbeitens herauszustellen.↩
159. Vgl.: Kapitel 2.1.1.↩
160. Vgl.: „Climbing on the shoulders of Giants.“; hier dargestellt in: Kapitel 5.1.1.↩
161. Titel der Originalausgabe: „On the Shoulders of Giants. A Shandean Postscript.“ (Robert K. Merton 1965).↩
162. Vereinte Dienstleistungsgesellschaft ver.di (http://portal.verdi.de/)↩
163. Präsident der Free Software Foundation Europe (http://www.fsfeurope.org/about/greve/)↩
164. Vgl.:
„Musikalische Viren“ (Goltzsch 1999), sehr früher Artikel, der sich mit der Gründung von GNUsic (s.u.) befaßt und alternative Perspektiven für Musiker aufzeigt.
GNUsic (http://gnusic.net); Projekt, welches sich seit 1998 für ein ähnliches wie in der Freien Software verwendetes Entwicklungsmodell engagiert und die Forderung „use/modify/redistribute“ für den künstlerisch-musikalischen Sektor formuliert.
Generell: Creative Commons (http://creativecommons.org/), heute die Herberge für den Gedanken des „some rights reserved“ in der Kunst. Unter Creative Commons ist eine Lizenz-Familie zu verstehen, die der jeweilige Lizenzgeber sehr genau an seine Wünsche bezüglich der Weiterverarbeitung und Verbreitung seines Werkes anpassen kann. Ab Januar 2005 wird es ein Unterprojekt geben, welches die Belange von Wissenschaftlern und Universitäten im Auge hat. Ziel ist es, „[T]o encourage scientific innovation by making it easier for scientists, universities, and industries to use literature, data, and other scientific intellectual property and to share their knowledge with others. Science Commons works within current copyright and patent law to promote legal and technical mechanisms that remove barriers to sharing.“ (Vgl.: http://science.creativecommons.org/)↩
165. Vgl.:
Steffen Richter: „Patente auf Software?“ (Richter 2003)
Übersicht zu heise-Artikeln (c't sowie Telepolis): http://patinfo.ffii.org/heise.html

Förderverein für eine freie informationelle Infrastruktur e.V.: http://swpat.ffii.org/
Sehr gute und v.a. „einfache“ Darstellung aus dem Alltag zu den „Gefahren“, sollte Software im Sinne des Europarat-Entwurfs patentierbar werden: http://webshop.ffii.org/↩

166. Non-profit Dachorganisation „Creative Commons“: http://creativecommons.org/↩
167. Entnommen aus: http://creativecommons.org/about/licenses/↩
168. Produktion aus Leipzig (2004); siehe http://www.route66-der-film.de/.↩
169. Schweizer Produktion aus dem Jahre 2003; siehe http://www.ch7.ch/.↩
170. Es wird der Technologie-Begriff herangezogen, da dieser – im Gegensatz zum Technik-Begriff – nicht nur auf die technischen Möglichkeiten zurückgreift, sondern zudem auch auf Produktions- und Organisationsprinzipien referiert.↩
171. Vgl.: Kapitel 2.1.5.↩
172. Zur Freiheit in weiteren Bereichen, welche ebenfalls von dem „frei“ in Freier Software inspiriert sind: siehe 5.1.1.↩
173. Für die Differenzierung zwischen Freier Software und Open Source Software, siehe Kapitel 5.4.3.↩
174. Historisch betrachtet ist der Ideologie-Begriff zumeist negativ belegt. Dies wird auf seine Verwendung unter Napoleon I. sowie bei Marx und Engels zurückgeführt. Dabei wurde der Begriff verwendet, um die Mechanismen einer herrschenden Schicht zur Erhaltung der ihnen vorteilhaften gesellschaftlichen Struktur zu beschreiben. In der Soziologie findet „Ideologie“ meist als Ideologiekritik Eingang in den Diskurs und ist auch hier negativ belegt.↩
175. Vgl.: „''Free software'' does not mean ''non-commercial''. A free program must be available for commercial use, commercial development, and commercial distribution. Commercial development of free software is no longer unusual; such free commercial software is very important. [Hervorh. i. Orig.; Anm. d. Verf.]“ (FSF 2004, S. 2)↩
176. http://www.mozilla.org/MPL/MPL-1.1-annotated.html↩
177. Originalwortlaut der Free Software Definition (FSF 2004, S. 1), man beachte die mit Null beginnende Zählung:
„The freedom to run the program, for any purpose (freedom 0).“
„The freedom to study how the program works, and adapt it to your needs (freedom 1). [...]“
„The freedom to redistribute copies so you can help your neighbor (freedom 2).“
„The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). [...]“.↩
178. In der Literatur häufige Benennung der Motivation des Mangels; ebenfalls gebräuchlich ist das Bild des „to scratch an itch“.↩
179. Vgl. dazu die Beschreibung früher Communities als Zirkel; Definiert in Fußnote 11.↩
180. Vgl. dazu das Buch von Erik Möller, Informatiker, Journalist und Mitarbeiter von Wikipedia: „Die heimliche Medienrevolution - Wie Weblogs, Wikis und freie Software die Welt verändern“ (Möller 2004): Möller geht auf die Beziehung von Macht und Medien unter der Perspektive des Internet als globalem, dezentralen Kommunikationsnetz sowie Freier Software ein. Insbesondere wird auf die Bedeutung der massiven Online-Zusammenarbeit von Freiwilligen eingegangen und mögliche Konsequenzen für die Content-Industrie.↩
181. Vgl.: Garstka 2003, S. 48ff.↩
182. Vgl.:
Thomas Fischermann in der ZEIT:
Fischermann 40/2004: „USA: Elektronische Wahlfälscher?“
Fischermann 45/2004: „Neue Maschienen, neues Ergebnis?“
Spiegel online:
Marc Pitzke 2004: „Zweifel am US-Wahlergebnis. Der Bezirk mit 139 Prozent Wahlbeteiligung.“
Wired (anerkanntes Technologie- und Internet-Kultur-Magazin):
Zusammenstellung wichtiger Artikel. Das News-Special läuft unter der Frage:
„E-voting gains a foothold at polling places in the U.S. and abroad, despite fears that the technology is flawed and votes easily manipulated. Is the voting booth still sacred? Does your vote still count?“ (Wired 2004; http://www.wired.com/news/evote/). Einzelne hervorzuhebende Beiträge:

> Kim Zetter 3/2004: „How E-Voting Threatens Democracy“
>
> Kim Zetter 10/2004: „Activists [u.a. Bev Harris, Autor von „Black Box Voting. Vote Tampering in the 21th Century.“ (Harris 2003); Anm. d. Verf.] Find More E-Vote Flaws“

Telepolis:
Alfred Krüger 11/2004: „Der große E-Voting Beta-Test“
Florian Rötzer 7/2003: „Das Problem mit den elektronischen Wahlsystemen und der amerikanischen Demokratie“
Wolf-Dieter Roth 10/2004: „USA: Die Qual der Wahl“↩
183. Vgl.: Friedman 2004.↩
184. Michael Moore: „Fahrenheit 9/11“: http://www.michaelmoore.com/warroom/f911notes/.↩
185. Für eine kurze Zusammenfassung der Kritikpunkte an als Fakten dargestellten Punkten in „Fahrenheit 9/11“, vgl.: Kopel 2004. Kopel geht auf 59 Aussagen des Films ein, welche nachweislich nicht der Wahrheit entsprechen.↩
186. „Ad Hoc Touch Screen Task Force Report“ des Secretary of State of California; unter:
http://www.ss.ca.gov/elections/taskforce_report_1.htm.↩
187. Vgl.: Harris 2003↩
188. http://www.blackboxvoting.org/↩
189. http://www.osce.org/documents/odihr/2004/11/3779_en.pdf↩
190. „The freedom to study how the program works, and adapt it to your needs [...]. Access to the source code is a precondition for this.“ (FSF 2004, S. 1)↩

191. „the freedom to improve the program, and release your improvements to the public, so that the whole community benefits [...]. Access to the source code is a precondition for this.“ (FSF 2004, S. 1)↩
192. Für den Fall Microsoft steht die „Shared Source Initiative“ einem erlauchten Kundenkreis mit hierarchisch gestuften Rechten zur Verfügung. Insgesamt ist davon auszugehen, daß ca. 10 % des gesamten Microsoft Code-Volumens aus „geschäftlichen Gründen“ nicht einsehbar ist (Vgl. zu der Thematik von Microsoft und Shared Source, Open Source und den jeweiligen Problemen: Krempl 2002). Die Darstellung der Initiative von Microsoft selber ist unter: http://www.microsoft.com/resources/sharedsource/Licensing/default.mspx zu finden.↩
193. Dieser Einwand besitzt eine klare praktische Begrenzung: Ein →Betriebssystem, auch wenn man es auf die wichtigsten Funktionen „abspeckt“, ist hinreichend groß und komplex, um eine komplette Überprüfung zu einem bestimmten Zeitpunkt unmöglich zu machen. Der, wenn auch kleine Vorteil von Freier Software – beispielsweise Linux, liegt darin, daß der →Quellcode zum Einen während der gesamten Evolution dieser Software offen liegt, zum Anderen weil der Kreis der „potentiellen Kontrolleure“ nicht künstlich klein gehalten wird. Einsicht in den Quellcode von beispielsweise Microsoft Windows XP zu bekommen ist eine sehr komplizierte Angelegenheit, mit sehr vielen Auflagen verknüpft und somit letztlich niemals „öffentlich“. Auch in diesem Punkt greift das Sprichwort von Eric Raymond: „Given enough eyeballs, all bugs are shallow.“ (Raymond 2001a, S. 30).↩
194. Die IG Kultur Österreich ist ein Netzwerk und eine Interessenvertretung der freien und autonomen Kulturarbeit in Österreich: http://igkultur.at/igkultur/organisation/1003760775.↩
195. http://www.debian.org/intro/about↩
196. http://www.debian.org/social_contract.en.html↩
197. http://www.opensource.org/docs/definition_plain.html↩
198. Der „Hacker Survey“, in Auftrag gegeben von der Boston Consulting Group, ist eine der wenigen repräsentativen empirischen Studien zum Thema Freie Software und Motivation. Ebenso wird eine ganze Batterie von interessanten Items bezüglich Freier Software abgefragt. Die Studie stützt sich auf einen Rücklauf von 684 Fragebögen, die einer Aktion von 2216 angeschriebenen Freien Software-Entwicklern entstammen. Vgl.: Lakhani et al. 2002.↩
199. Schöpfer von TEX, einer Freien Software unter welcher ca. 95 % der Arbeiten in Mathematik und Physik verfaßt werden.↩
200. Vorläufige Resultate des empirischen Teils der Studie in Luthiger (2004). Einen Überblick über die Studie und ihrer Methodik findet sich unter: http://www.isu.unizh.ch/fuehrung/blprojects/FASD/.↩
201. Vgl.: Williams 2002, S. 15f.↩
202. Vgl.: Schulze 2004, S. 5.↩
203. Vgl.: Raymond 2001a, S. 23.↩
204. Popitz datiert diesen Bruch vage auf das späte Neolithikum, der Epoche der Menschheitsgeschichte, auf die man heute den Übergang von Jäger- und Sammlerkulturen zu Hirtentum und dem Ackerbau datiert. Konkrete Datierungen fallen je nach Kulturraum unterschiedlich aus; für Mitteleuropa wird die Zeitspanne von ca. 5500 bis 1800 v. Chr. angenommen.↩
205. Vgl.: Die vier Freiheiten der →GNU General Public License (GPL) ; u.a. dargestellt in Kapitel 5.4.1.↩
206. Vgl.: An erster Stelle dieser „Folge-Motivationen“ steht Community-Gedanke (siehe Kapitel 5.3). Daraus können sich wiederum weitere Motivationen entwickeln (siehe Kapitel 5.1, 5.2, 5.4).↩
207. „Improve skill“ wird im Hacker Survey der Boston Consulting Group als zweitwichtigste Motivation für ein Engagement in Freier Software aufgeführt (Lakhani et al. 2002, S. 12). Zum selben Ergebnis kommt Lakhani und Wolf in einer späteren Studie (Vgl.: Lakhani/Wolf 2003, S. 13).
In der FLOSS-Studie wird von „learn and develop new skills“ als der wichtigsten Motivation für ein Engagement in Freier Software berichtet (Ghosh et al. 2002, S. 46).↩
208. Vgl. zur Einschätzung der niedrigen Einstiegsbarrieren für ein Engagement in Freier Software die Aussagen des ehemaligen Microsoft Executive Officers Nat Brown, der dies – als Kenner des „proprietären Lagers“ – für den Überlegenheitsfaktor gegenüber dem proprietären Modell hält. (in: Becker 2004, S. 1)↩
209. Vgl.: Harald Welte's Engagement für das connection tracking [dt.: Verbindungsverfolgung] im →TCP/IP-Stack der Entwicklerversion 2.3 des Linux-Kernels. (Welte 2004, S. 3)↩
210. Zur Unterscheidung zwischen „einfachen“ und „höheren“ Motivationen siehe Kapitel 6.2.2.↩
211. siehe zusammenfassend auch Kapitel 6.2.1.↩
212. Für die Institutionalisierung sei beispielhaft die →Free Software Foundation genannt, welche inzwischen auch auf europäischer Ebene eine Niederlassung besitzt. Momentan ist der Interviewpartner Werner Koch Vize-Kanzler der europäischen Niederlassung (Free Software Foundation Europe (FSFE); http://www.fsfeurope.org). Die FSFE versteht sich als Anwalt Freier Software sowie als „Kompetenzzentrum für Politiker, Anwälte und Journalisten [...], um die juristische, politische und gesellschaftliche Zukunft Freier Software zu gewährleisten.“ (FSFE 2004, S. 1). Dies berichtet auch Interviewpartner Harald Welte (Welte 2004, S. 4f).
Am deutlichsten tritt die Institutionalisierung in der aktuellen Debatte um die Gesetzesänderung zu „Software-Patenten“ hervor. Hier wird auf nationaler wie europäischer Ebene ein regelrechter Kampf gegen die „Patentierung von Ideen“ geführt. Ein Kampf, welcher inzwischen auch von der Seite der Fraktionen des Bundestages unterstützt wird. Diese Diskussion um Software-Patente wurde für die vorliegende Arbeit versucht auszuklammern, da sie in ihrer Dimension hier nicht mehr adäquat behandelt werden könnte.↩
213. Von Robert K. Merton geprägter Begriff, der den Peer Review-Prozeß in wissenschaftlicher Forschung umschreibt. Dabei sind die Prinzipien der gegenseitigen Überprüfung, Kritik und Fortschreibung von wissenschaftlichen Ergebnissen auch auf das Freie Software-Modell übertragbar. (Vgl.: Merton 1974, S. 267ff)↩
214. Vgl.: Hardin 1968.↩

ABOUT JOURNALTHING
journalthing is a journal platform.
All content is freely accessible at https://journalthing.thms.de, with four online and print issues published annually.

Stay informed by emailing to journalthing-subscribe@journalthing.thms.de to receive alerts for new releases.