Motivation und Organisation
von
Open Source Projekten

Projektarbeit von Caspar Clemens Mierau
Seminar: Tragik des Cyberspace

Bauhaus-Universität Weimar
Studiengang: Medienkultur

Dozent: Knuth Baumgärtel

Sommersemester 2002

- Inhaltsverzeichnis -

Einleitung

In den letzten Jahren lässt sich in Politik und Wirtschaft eine hitzige Diskussion um ein Thema beobachten, das eigentlich gänzlich unpolitisch wirkt: Software. Die Debatte dreht sich dabei um das Für und Wider des Einsatzes Freier Programme. Von der kommerziellen Softwareindustrie verschrien, von merkwürdig anmutenden Jüngern propagiert, scheiden sich an dieser Frage die Gemüter.

Für Außenstehende ist es unverständlich, dass Tausende von Programmierern weltweit vernetzt, ohne Bezahlung gemeinsam an Softwareprojekten arbeiten, diese kostenlos vertreiben und überdies noch Support und Öffentlichkeitsarbeit leisten. In einer marktwirtschaftlich organisierten, Gesellschaft, die mit Arbeitslosigkeit zu kämpfen hat, widerstrebt der scheinbare Verzicht auf Verfügungsrechte und Entlohnung den Vorstellungen der Ökonomie.

Die folgende Arbeit wird sich mit der Frage auseinandersetzen, welche Motive Personen dazu bewegen, ihre Zeit der Entwicklung freier Software zur Verfügung zu stellen und wie eine Koordination von großen Software-Projekten möglich wird, wenn das wichtigste wirtschaftliche Motivationsinstrument wegfällt: Geld. Dabei soll hier weniger auf ideologisch geprägte Quellen geachtet werden, als vielmehr der Versuch unternommen, das Phänomen Open Source Entwicklung durch eine Konfrontation mit anerkannten Paradigmen zu untersuchen.

Entstehung der Software-Industrie

Um die Entwicklung einer Open Source Szene als ganzheitlichen Prozess verstehen zu können, der nicht nur eine Reaktion auf kommerzielle Software ist, muss ein Blick auf die Geschichte der Software-Industrie geworfen werden. Zuvor jedoch eine Erklärung des Begriffes "Quellcode", da er für das Verständnis dieser Arbeit von essentieller Bedeutung ist:

Ein Quellcode (Quelltext, Source Code) ist das in einer menschenlesbaren Programmiersprache geschriebene Programm, das durch den Vorgang des Kompilierens in einen von Computern ausführbaren Objektcode übersetzt werden muss. Kommerzielle Software wird heute üblicherweise als Objektcode ausgeliefert, der nicht verändert und untersucht werden kann und darf[1]. Metaphorisch könnte man sagen, dass Software, die nur im Objektcode ausgeliefert wird, sich wie ein Auto mit abgeschlossener Motorhaube verhält: Es kann zwar fahren und somit seinen Zweck erfüllen, für den Käufer bleibt es jedoch ein Rätsel, was sich unter der Motorhaube verbirgt und wie man es im Fall einer Panne reparieren kann[2].

IBM-kompatible PCs

In den 60er und 70er Jahren, den Anfängen der Computerwirtschaft, füllten Rechneranlagen noch ganze Räume aus und wurden in sehr kleinen Stückzahlen vertrieben. Da Computerprogramme ohne Hardware und Hardware ohne

[1] Stichwort "Quellcode" in "Geoinformatik Lexikon", elektronisch veröffentlicht unter http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=1412 [Stand: 30.09.2002]

[2] Der Vergleich mit der verschlossenen Motorhaube findet sich recht oft in der Linux-Szene, siehe zum Beispiel das Interview mit dem Autor Peter Ganten, elektronisch veröffentlicht unter: http://www.tachauch.de/background/computer/3003_interview.html [Stand: 30.09.2002]

Computerprogramme keinen Wert hatten, wurde Software frei als Quelltext vertrieben. Hardwarehersteller förderten die Initiativen der Anwender, Ihre Programme untereinander auszutauschen und weiterzuentwickeln, um die Attraktivität der eigenen Produkte durch ein erhöhtes Angebot an Software zu steigern[3]. Eine positive Wirkung hatten Time-Sharing-Systeme wie IBMs SHARE, die zu diesem Zweck den Zugriff auf Dateien von verschiedenen Computern ermöglichten (Und damit nicht zuletzt die Grundlagen des Internets schufen[4]).

Durch die Monopolstellung IBMs bei Großrechnern und den gemeinsamen Vertrieb von Support, Soft- und Hardware gab es kaum Wettbewerb und so wurde 1969 ein Antitrust-Verfahren gegen den Hardwarehersteller in den USA eingeleitet (Es sollte 13 Jahre dauern und danach stillschweigend eingestellt werden[5]). Bis zu diesem Zeitpunkt konnte man nicht von einer Softwareindustrie sprechen, da es in diesem Bereich keine oder nur wenig ernstzunehmende Konkurrenz gab. 1981 jedoch führte IBM den ersten Mikroprozessor-PC ein, dessen Architektur überraschenderweise offengelegt wurde. Die Prozessoren wurden von Intel, das Betriebssystem DOS von Microsoft eingekauft und somit die Produktion ausgelagert. Warum IBM das hauseigene "Bundling" aufgegeben hat, ist bis heute unklar. Sicher ist jedoch, dass durch die billigen IBM-kompatiblen PCs mit einer offengelegten Architektur einer Softwareindustrie wie wir sie heute kennen erst die Entstehung ermöglicht wurde[6].

---

[3] GRASSMUCK, Volker (2000): "Open Source - Betriebssystem für eine freiheitliche Gesellschaft", elektronisch veröffentlicht unter http://waste.informatik.hu-berlin.de/Grassmuck/Texts/OSS-Tutzing-5-00.html [Stand: 22.09.2002]

[4] JAEGER, Till und Dr. METZER, Axel (2002): "Open Source Software", München (Verlag C.H. Beck), S. 8

[5] STIELER, Dr. Wolfgang: "(Un)endliche Geschichte" in "c't magazin für computer technik", Ausgabe 20/2002 (23.9.2002), Hannover (Verlag Heinz Heise), S. 3

[6] JAEGER/METZER (2002), S. 9

Unix

Parallel zu dieser Entwicklung entstand das Betriebssystem Unix, dass die Freie Software Szene nachhaltig prägte. 1969 versuchte ein Mitarbeiter der amerikanischen Telefongesellschaft AT&T ein Spiel, das er auf einem Lochkartenrechner entwickelt hatte, auf eine Großrechenanlage zu portieren, um mehr Performance (Geschwindigkeit) zu erreichen[7]. In wenigen Jahren entstand aus diesem Vorhaben das heute unter dem Namen "Unix" bekannte Betriebssystem, dass besonders an Universitäten beliebt wurde. Verantwortlich hierfür war vor allem der Fakt, dass es AT&T als staatlich kontrolliertem Telefonmonopolist verboten war, sich in anderen Wirtschaftsbereichen zu engagieren und so Unix-Quelltexte zum Selbstkostenpreis von USD 50,- an Universitäten abgeben wurden[8].

Nachdem AT&T 1984 durch einen Kartellrechtsprozess in mehrere, konkurrierende Unternehmen zerschlagen wurde, konnte die kommerzielle Ausnutzung von Unix intensiviert werden – die Lizenzgebühren stiegen erheblich. Bereits weit verbreitete Unix-Weiterentwicklungen wie BSD sahen sich vor Lizenzproblemen, da sie Quelltexte des AT&T-Unix übernommen hatten. Um sich aus dieser Misslage zu befreien, entstanden die Open Source BSD-Versionen NetBSD, FreeBSD und OpenBSD, die um den AT&T Code bereinigt wurden und noch heute weiterentwickelt werden[9], sowie zeitlich das freie GNU/Linux.

---

[7] KAHLHAMMER, F.: "Betriebssystem Linux – Die Entstehung des Betriebssystems", elektronisch veröffentlicht unter http://www.linux-praxis.de/linux1/geschichte.html [Stand: 22.09.2002]

[8] GRASSMUCK, Volker (2002): "Freie Software: Zwischen Privat- und Gemeineigentum", Bonn (Bundeszentrale für politische Bildung), elektronisch veröffentlicht unter http://freie-software.bpb.de/Grassmuck.pdf [Stand: 22.09.2002], S. 212

[9] siehe http://www.freebsd.org, http://www.netbsd.org, http://www.openbsd.org

Es kann festgehalten werden, dass Open Source Software keine revolutionäre Idee ist. Im Gegenteil – Freie Software wurde von der Geburtsstunde der Computertechnologie an verbreitet und sogar von der Wirtschaft gefördert um eigene Gewinne zu maximieren. Erst zu Beginn der 80er Jahre veränderte sich die Situation durch die Kommerzialisierung des Unix-Betriebssystems und die Geburtsstunde des IBM-kompatiblen PCs.

Freie Software / Open Source Software

Der Begriff der Freien Software existiert seit der Mitte der 80er Jahre, die mittlerweile geläufige Bezeichnung "Open Source Software" erst seit dem Jahr 1998[10]. Den Bemühungen der freien Softwareszene ist es zu verdanken, dass durch kontinuierliche Selbstreflexion diese Begriffe heute recht genau definiert werden können.

Die wohl älteste und verlässlichste Definition Freier Software liefert die bereits 1985 von Richard Stallmann gegründete "Free Software Foundation" (FSF), deren Hauptanliegen die Entwicklung und verbreitete Nutzung freier Software sind[11]. So heißt es in einer prägnanten Kurzformel: "Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software"[12]. Hervorgehoben wird der Fakt, dass das Wort "frei" sich nicht auf ein "kostenlos" wie in "Freibier" kürzen lässt, sondern wie das "frei" in der Floskel "freie Rede" zu verstehen ist.

[10] JAEGER/METZER (2002), S. 1
[11] Selbstauskunft der "Free Software Foundation", elektronisch veröffentlicht unter http://www.fsf.org/fsf/fsf.html [Stand: 09.17.2002]
[12] "The Free Software Definition" der "Free Software Foundation", elektronisch veröffentlicht unter http://www.fsf.org/philosophy/free-sw.html [Stand: 09.17.2002]

Man kann die wichtigsten Aussagen der FSF-Definition freier Software wie folgt untergliedern:

- Die der Software zugrunde liegende Lizenz muss eine **uneingeschränkte Weitergabe** der Software ermöglichen, ohne dass hierfür Lizenzgebühren anfallen.
- Die Software muss im **Quellcode** vorliegen und darf im Quellcode oder in kompilierter Form weitergegeben werden. Wird sie in kompilierter Form angeboten, muss der Quellcode frei erhältlich sein.
- Die Lizenz muss ermöglichen, dass die Software **weiterentwickelt** und die veränderte Version unter die gleichen Lizenzbedingungen gestellt werden kann.
- Die Lizenz darf keine Personen oder Personengruppen von der Nutzung ausschließen.

Jaeger/Metzger fassen zusammen: "Der entscheidende Unterschied zwischen Freier Software und herkömmlicher Software besteht damit in der umfassenden Einräumung urheberrechtlicher Nutzungsrechte, die das freie Kopieren, Bearbeiten, Untersuchen und Verbreiten ermöglichen"[13]. Verfügungsrechte, unter denen man "das Recht, physische Güter oder Leistungen zu gebrauchen und Nutzen aus ihnen zu ziehen sowie das Recht von anderen Personen ein bestimmtes Verhalten zu fordern"[14] versteht, werden hier folglich nicht dazu genutzt, Lizenzgebühren rechtlich abzustecken, sondern weitergehende Nutzungs- und Entwicklungsmöglichkeiten zu sichern.

Der Begriff "free software" wurde 1998 um das Synonym "open source software" erweitert. Dies geschah vor allem durch die

[13] JAEGER/METZER (2002), S. 3
[14] RICHTER, Rudolf (1999): "Neue Institutionenökonomik", Tübingen (Mohr Siebeck), S. 5

Initiative einer Marketing-Offensive der Freien Software Szene. Das Wort "frei" wurde von der Wirtschaft abgelehnt, da es zu leicht mit der "Kultur des Verschenkens und der Geschäftsfeindlichkeit" verbunden werden kann[15]. Nach einer Ankündigung Netscapes, die Quelltexte ihres Browsers offenzulegen, wurde am 3.2.1998 in Palo Alto (Kalifornien) die "Open Source Initiative" gegründet und beschlossen, fortan den Term "Open Source Software" zu verwenden[16]. In der Freien Software Bewegung führte dies zu einer Spaltung, da man einen Paradigmenwechsel weg von der "Freien" Software hin zu einer Software, die zwar im Quelltext vorliegt aber nicht der Definition nach lizenzrechtlich frei ist, sieht. Diese Diskussion soll an hier nicht weiter beachtet werden. Im Folgenden werden die Begriffe "Freie Software" und "Open Source Software" daher ohne Unterschied gebraucht.

Geld als ungeeignetes Tauschmittel

Ohne Geld sind heute wirtschaftliche Transaktionen undenkbar. Geld ist das anerkanntes Zahlungs- und Tauschmittel. Um zu verstehen, warum Geld in der Freien Software Szene weniger Beachtung findet als in der kommerziellen, soll nun der Versuch unternommen werden, die Unzulänglichkeit des Geldes als universelles Tauschmittel darzulegen.

Möchte eine Person ein materielles oder immaterielles Gut weitergeben, ohne es zu verschenken, muss sie es gegen ein anderes Gut eintauschen. In einer arbeitsteiligen Wirtschaft ist es jedoch nur selten möglich, Güter direkt gegeneinander einzutauschen. Man geht davon aus, dass Geld aus dem Bedürfnis

[15] JAEGER/METZER (2002), S. 3
[16] "History of the OSI", elektronisch veröffentlicht unter http://www.opensource.org/docs/history.html [Stand: 22.09.2002]

nach einem Wertmaßstab für die quantitative Bedeutung von verschiedenartigen Gütern entstanden ist. So macht das übergeordnete Tauschmittel Geld materielle und immaterielle Güter für einen Tausch vergleichbar[17].

Dem Medium Geld kommen hierbei drei Funktionen zu, die bereits im 19. Jahrhundert von der klassischen Ökonomie dargestellt wurden[18]:

- Geld ist ein **Tauschmittel**, das einen Tauschwert, aber keinen Gebrauchswert hat.
- Geld dient als **Wertmaßstab**, weil es Güter und Dienstleistungen sowie Produktionsfaktoren, die völlig heterogen sind durch den Bezug auf die ökonomische Größe des Nutzens vergleichbar macht.
- Geld dient als **Wertaufbewahrungsmittel**, weil man bei einem Rücktausch gegen Güter oder Leistungen keinen Wertverlust erleidet.

Nach der Geldtheorie besteht der Nutzen des Geldes in einer arbeitsteiligen Tauschwirtschaft darin, dem Einzelnen Liquidität (Zahlungsfähigkeit) zu gewährleisten. Das Liquiditätsbedürfnis ergibt sich aus der Unsicherheit der Individuen über mögliche ökonomische Transaktionen[19]. Es steht somit außer Frage, dass ein Grundbestreben in der arbeitsteiligen Tauschwirtschaft in der Beschaffung von Geld liegt, um die eigene Liquidität sichern zu können. Die politische Brisanz des Themas Arbeitslosigkeit bestärkt diese These.

---

[17] Stichwort "Geld" aus Brockhaus/Xipolis

[18] PARSONS, Talcott (2000): "Sozialstruktur und die symbolischen Bildmedien" in "Kursbuch Medienkultur", Stuttgart (DVA), S. 35

[19] Stichwort "Geld" aus "Brockhaus - Die Enzyklopädie: in 24 Bänden", elektronisch veröffentlicht unter http://www.xipolis.de [Stand: 30.09.2002]

Trotz seiner universellen Eigenschaften für ökonomische Tauschvorgänge scheint Geld nicht immer ein geeignetes Tauschmedium zu sein. So gibt es "viele andere Tauschvorgänge zwischen Menschen, die sich nicht durch Geld vermitteln lassen"[20]. Die Tauschtheorie bildet hier eine Schnittstelle zwischen Ökonomie und Soziologie. So betrachten Theoretiker wie Peter M. Blau und George C. Homans die Ordnung sozialer Interaktionen in weiten Bereichen als Tauschvorgänge, denen die Norm der Reziprozität (Gegenseitigkeit) gemeinsam ist[21].

Dass für gegenseitige Beziehungen wie Liebe und Freundschaft Geld kein geeigneter Wertemaßstab bzw. kein geeignetes Tauschmittel sein kann, muss hier nicht näher erläutert werden. Für Dienstleistungen hingegen ist man eher geneigt, Geld als universelles Tauschmittel einzusetzen. So sichert ein Arbeitsvertrag dem Arbeitnehmer zu, dass der Arbeitgeber für die Dienstleistung des Arbeitnehmers eine vereinbarte Vergütung zahlt[22]. Kann Geld als Motivationsfaktor dennoch im Dienstleistungsbereich an die Grenzen seiner Einsetzbarkeit gelangen?

Allein die Tatsache, dass 1995 die kumulierten Arbeitsstunden ehrenamtlicher Mitarbeiter im westdeutschen Nonprofit Sektor 1,26 Millionen Vollzeitarbeitsplätze ergaben[23], stützt die Vermutung, dass auch Dienstleistungen nicht immer mit Geld vergütet werden müssen oder können. Das Wort "ehrenamtlich" (um der "Ehre" Willen) kann als Indiz für einen von der Geldtheorie abgelösten Tauschprozess gesehen werden. Es scheint Motive für Arbeit zu geben, die nicht zwangsläufig auf die Beschaffung von Geld

[20] PARSONS (2000), S. 36
[21] JOAS, Hans (Hrsg.)(2001): "Lehrbuch der Soziologie", Franfurt/Main (Campus Verlag), S. 103
[22] RICHTER (1999), S. 149
[23] BADELT, Christoph (Hrsg.)(2002): "Handbuch der Nonprofit Organisation", Stuttgart (Schäffer-Poeschel), S. 29

ausgerichtet sind. Nach einer Studie liegen die Beweggründe für ehrenamtliche Arbeit neben der Tätigkeit an sich in einem Wunsch nach sinnvoller Nutzung von Freizeit, Kontaktbedürfnissen, dem Sammeln von Erfahrungen und im Lernen[24].

Trotz der Ansicht der individualistischen Eigentumslehre, die im Menschen "ein selbstsüchtiges Geschöpf, das sich selbst mehr liebt als jeden anderen"[25] sieht, muss ehrenamtliche Arbeit also nicht einer sozialistischen Lehre zugeordnet werden. Das unentgeltliche Arbeiten erfolgt nicht zwangsläufig aus einer selbstlosen Lebenseinstellung, sondern vielmehr aus individualistischen, jedoch nicht-monetären Bedürfnissen.

Motivation

Wie oben bereits verdeutlicht wurde, ist der Erwerb von Geld nicht immer Hauptbeweggrund für das Erbringen von Dienstleistungen. Da Mitarbeiter an Open Source Projekten in der Regel "for free" arbeiten, soll nun untersucht werden, welche Motive sie dazu bewegen, Ihre Arbeitskraft dennoch zur Verfügung zu stellen. Eine Beantwortung dieser Frage kann dank erster statistischer Erhebungen durch empirische Erkenntnisse gestützt werden.

Eine Studie der Universität Kiel kommt zu dem Schluss, dass das Engagement bei der Entwicklung von Linux (dem wohl bekanntesten Open Source Projekt) durch Motive verursacht wird, die denen sozialer Bewegungen wie Menschenrechts- und Friedensbewegungen ähnlich sind[26]. Mag man jetzt vermuten, dass

[24] BADELT (2002), S. 311
[25] RICHTER (1999), S. 79
[26] HERTEL, Guido: "Motivation of Software Developers in Open Source Projects: An Internet-based Survey of Contributors to the Linux Kernel" (Version: Sept/11/2002), 2002, noch nicht veröffentlicht, S. 29

altruistische Züge hier die Oberhand haben, wird man von der folgenden Zusammenfassung überrascht. Die wichtigsten Motive für Linux-Entwickler sind demnach:

1. eine **generelle Identifikation** als Linux-Benutzer
2. eine **speziellere Identifikation** als Linux-Entwickler
3. **pragmatische Motive** wie die Verbesserung eigener Software oder Vorteile im Berufsleben
4. **Norm-orientierte Motive** wie soziale Beziehungen zu Familie, Freunden und Kollegen
5. **soziale und politische Motive**, der Wunsch, freie Software zu unterstützen
6. **hedonistische Motive** wie die Freude am Programmieren

Eine altruistische Komponente kann hier höchstens in die Rubrik der sozialen und politischen Motive fallen und somit nicht Hauptbeweggrund für Programmierer sein. Weitaus wichtiger sind Elemente wie Identifikation, pragmatische Motive, Hedonismus und einstweilen auch Geld.

Identifikation

Eine überraschend wichtige Komponente in der Open Source Szene ist die **Identifikation mit der Community** (Gemeinschaft), die gerade unter Linux Entwicklern sehr groß ist[27]. Diese Beobachtung wird gestützt durch das Ergebnis einer Studie der "Boston Consulting Group" (BCG) in Zusammenarbeit mit dem "Open Source Development Network" (OSDN), die "strong identification" als eines der Haupterkenntnisse ihrer Umfrage anführt[28]. Trotz virtuell vermittelter Kommunikation nehmen soziale Bindungen

[27] HERTEL (2002), S.29
[28] LAKHANI, Karim (2002): "The Boston Consulting Group Hacker Survey" (Version 0.73), elektronisch veröffentlicht unter http://www.osdn.com/bcg/BCGHACKERSURVEY-0.73.pdf [Stand: 20.09.2002], S. 3

eine wichtige Stellung in der Open Source Szene ein, die als Antrieb für das Engagement nicht unterschätzt werden sollten.

Entwicklungszeit und Innovation

Durch die Offenlegung von Quelltexte wollen Freie Software Projekte nicht einfach die Wohlfahrt fördern, sondern vielmehr pragmatische Strategien verfolgen. Ein wichtiges Ziel ist die effiziente Verkürzung der **Entwicklungszeit**. "Release early, release often", wie es Eric Steven Raymond in seinem viel zitierten Aufsatz "The Cathedral and the Bazaar" fordert[29], verdeutlicht, dass eine frühe und häufige Veröffentlichung von Quelltexten eine zentrale Strategie bei der Entwicklung Freier Software ist. Erst durch Veröffentlichung wird **Kollaboration** ermöglicht. Linus Torvalds, der Vater und Namensgeber des freien Betriebssystems Linux, veröffentlichte seine Quelltexte einst, um andere Programmierer für eine Mitarbeit gewinnen zu können.

Während eine große Zahl an Nutzern und Entwicklern einerseits Entwicklungszeiten verkürzen kann, bietet sie auch die Möglichkeit, **Innovationen** zu erleichtern. Einzelne Personen sind durch ihre begrenzte Rationalität nur schwer in der Lage, Innovationen hervorzubringen. Erst mehrere, sich ergänzende Individuen, können gemeinschaftlich Innovationen hervorbringen und haben somit die Chance, durch ihre gemeinschaftliche Arbeit die Wohlfahrt zu steigern. Auf den Punkt gebracht lautet die These: "Innovation is often a process to which several actors with complementary capabilities contribute. Bringing these actors together is often welfare-improving, since none of them has

[29] RAYMOND, Eric Steve (2002): "The Cathedral and the Bazaar" (Version 3.0), elektronisch veröffentlicht unter http://www.tuxedo.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html [Stand: 21.09.2002]

sufficient knowledge or information to produce the innovation on their own"[30].

Qualität

Die Sicherung der Qualität erfolgt nach dem Prinzip "vier Augen sehen mehr als zwei", oder wie Raymond es ausdrückt "Given enough eyeballs, all bugs are shallow"[31]. Bugs (Fehler) im Programmcode werden von vornherein zugestanden, dafür sollen sie durch eine **offene Kommunikation** schneller gefunden und beseitigt werden. Für den einzelnen Entwickler, der oft auch Nutzer seiner eigenen Entwicklung ist, bedeutet dies eine qualitativ wertvollere Software, die sicherer und stabiler läuft. "Release early, release often" bedeutet hier, nicht auf ein vermeintlich fertiges Produkt zu warten, sondern bewusst Unfertiges freizugeben, um die **Qualitätssicherung** in den Prozess der Programmentstehung einzubauen. Hilfreiches Feedback ist einerseits von anderen Programmierern, die den Programmcode verstehen und kommentieren können, als auch von Endanwendern, die das Programm nutzen, zu erwarten.

Netzeffekte und Standards

Einige Güter werden erst dann für den Besitzer wertvoll, wenn mehrere Nutzer das gleiche (oder kompatible) Gut benutzen. Man spricht hier von "positiven Netzwerkexternalitäten". Ein klassisches Beispiel ist das Telefonnetz – je mehr Personen über einen Fernsprechapparat mit Anschluss an das Netz verfügen, desto wertvoller ist der eigene Anschluss für jeden Einzelnen, da durch

[30] HARHOFF, Dietmar, HENKEL, Joachim und von HIPPEL, Erich (2000): "Profiting from voluntary information spillovers: How users benefit by freely revealing their innovations", elektronisch veröffentlicht unter http://opensource.mit.edu/papers/evhippel-voluntaryinfospillover.pdf [Stand: 21.09.2002]

[31] RAYMOND (2002)

die höhere Verbreitung mehr Individuen erreicht werden können[32]. Gerade Software ist auf positive Netzeffekte angewiesen. Bis auf wenige Ausnahmen wird ein Programm erst dann nützlich, wenn man die mit ihm produzierten bzw. verarbeiteten Informationen mit anderen Nutzern tauschen kann. Es ist heute zum Beispiel kaum denkbar, mit einer Textverarbeitung zu arbeiten, deren Dokumente nicht per Diskette oder Email an andere Personen übergeben werden können.

Für die Nutzbarkeit eines Software-Systems ist somit eine "kritische Masse" erforderlich, die den mindestens benötigten Verbreitungsgrad der gleichen oder kompatiblen Software angibt. Zur Erreichung dieser Masse kann es sinnvoll sein, offene Standards zu schaffen. "If you own valuable intellectual property but need to gain critical mass, you must decide whether to promote your technology unilaterally in the hope that it will become a de facto standard that you can tightly control, or to make various 'openness' commitments to help achieve a critical mass"[33] schreiben Shapiro/Varian in ihrem Klassiker "Information Rules".

Ein Exempel aus der kommerziellen Software-Industrie ist das vom Fraunhofer Institut entwickelte Audio-Format mp3, das sich in sehr kurzer Zeit einer weiten Verbreitung im Internet erfreute, da Programme zur Erstellung und zum Abspielen von mp3s kostenlos erhältlich waren. Erst 1998 begann das Institut damit, seine Lizenzanspruche geltend zu machen und Lizenzgebühren zu verlangen bzw. Mahnungen zu verschicken[34]. Durch Offenlegung

---

[32] RICHTER (1999), S. 317

[33] SHAPIRO, C./ VARIAN, H. (1998): "Information Rules: A Strategic Guide to the Network Economy", Boston, S. 16

[34] RÖTTGERS, Janko (2001): "Neues Jahr, neuer Codes" in "Telepolis", elektronisch veröffentlicht unter http://www.heise.de/tp/deutsch/inhalt/musik/4617/1.html [Stand: 22.09.2002]

der Technologie hatte man eine schnelle und weite Verbreitung erreicht, auf der nun die kommerzielle Ausnutzung aufsetzen konnte.

Für viele Nutzer hinterließ dieses Vorgehen einen unangenehmen Eindruck, da sie nicht bereit waren, für etwas zu zahlen, das ihnen bis dahin kostenlos schien. So setzte sich die Open Source Gemeinschaft mit einem eigenen Audio Format zur Wehr. "Vorbis" ist eine Patent-freie Open Source Alternative, die einen verbesserten Funktionsumfang gegenüber mp3 haben soll[35]. Die notwendige kritische Masse kann dieses Format aber nur dann erreichen, wenn es bereits mit Playern oder gar Betriebssystemen ausgeliefert wird. Mp3 wird aufgrund seines hohen Bekanntheitsgrades trotz der Lizenzgebühren weiterhin von vielen Webseiten wie mp3.com und besonic.de[36] eingesetzt. Nach einer Aufnahme von Vorbis in das weit verbreitete Programm Winamp und die Ankündigung des Multimedia-Riesen Realnetworks, Vorbis zu unterstützen[37], erfährt das Format eine steigende Aufmerksamkeit und somit positive Netzeffekte, die die Distanz zur kritischen Masse verringern.

Die Motivation für der Entwicklung von Vorbis liegt also nicht in der selbstlosen Aufopferung der Programmierer für das Allgemeinwohl, vielmehr sollen Lizenzgebühren umgangen und ein frei nutzbares Audio-Format geschaffen werden. Unter diesem Gesichtspunkt ist das Offenlegen von Innovationen eine individualistische Strategie. Harhoff et al fassen als "Setting a standard advantageous to the user innovator"[38] zusammen.

---

[35] siehe "Vorbis FAQ", elektronisch veröffentlicht unter http://www.vorbis.com/faq.psp [Stand 22.09.2002]

[36] siehe auch http://www.mp3.com und http://www.besonic.de

[37] Artikel "Real will freies Audioformat Ogg Vorbis unterstützen", elektronisch veröffentlicht unter http://www.heise.de//newsticker/data/vza-25.07.02-000/ [Stand: 22.09.2002]

[38] HARHOFF et al (2000), S.2

Der Nutzer als Entwickler und umgekehrt

Aus den bereits genannten Faktoren ergibt sich die Einsicht, dass Open Source Entwickler oft an Programmen arbeiten, **die sie selbst auch beruflich oder privat nutzen**. Über 30% der Befragten geben in der Studie der BCG/OSDN an, dass ihre Motivation in der Erweiterung der "functionality" der bearbeiteten Projekte liegt. Eine gemeinsame Entwicklung mit anderen Open Source Programmierern ist folglich auch in diesem Sinne eine stark individualistische Komponente, da eigene Innovationen veröffentlicht werden, um sie überprüfen zu lassen und gegen andere Innovationen einzutauschen[39].

Titel und Status

In der Open Source Szene nehmen Titel eine besondere Stellung ein. Normale Firmen-Karriereleitern gelten zwar als nebensächlich, die **Anerkennung in der Community** aufgrund eigener Fähigkeiten hingegen ist eine nicht zu unterschätzende Motivation[40]. Während der **Titel** des "stellvertretenden Geschäftsführers" wenig Bewunderung auslöst, kann ein "Chefentwickler" mehr beeindrucken. Soziologisch betrachtet handelt es sich hierbei um einen "erworbenen Status"[41], der durch eigene Anstrengungen erreicht wurde, sich jedoch immer wieder beweisen muss.

George N. Dafermos, der die Managementstrukturen des Linux-Projektes genauer untersucht hat, fasst zusammen, dass selbst Linus Torvalds als "Leader" mit seinen Entscheidungen logisch nachvollziehbare Schritte gehen muss, da andere Entwickler seine

[39] vgl. HARHOFF et al (2000)
[40] vgl. PAVLICEK S. 147f
[41] JOAS (2001), S. 111

Vorschläge herausfordern ("challenge"[42]) können. Ein Status und Anerkennung in der Open Source Gemeinschaft wird nach BCG/OSDN von über 17% der Befragten als Motivationsgrund angegeben. Überdies hat der Status einen praktischen Nutzen: Immer mehr Softwareunternehmen umwerben ganz gezielt Personen, die sich in der Szene durch ihre Fähigkeiten beweisen konnten[43].

Hedonismus

Hedonismus bei Programmierern kann von Außenstehenden nur schwer erkannt oder nachvollzogen werden. Man muss akzeptieren, dass für viele Software-Entwickler das Programmieren selbst eine Befriedigung darstellt. Die Studie der BCG/OSDN gibt an, dass rund 45% aller Befragten die Gruppe der "Hobbyists" das Programmieren als "Intellectually stimulating" und wie das Schreiben von Poesie oder Musik empfinden[44].

Vergleicht man das Hobby des Software Entwicklers mit dem eines Briefmarkensammlers ist der größte Unterschied, dass eine Briefmarkensammlung nur der sammelnden Person nutzt, während Freie Software auch anderen Personen dienen kann, da sie kopierbar ist. In diesem Sinne kann selbst Hedonismus, eines der egoistischsten Motive, der Wohlfahrt dienen.

Geld

Trotz der bisherigen Überlegungen muss erwähnt werden, dass nicht alle Entwickler Freier Software ihre Arbeitskraft völlig unentgeltlich zur Verfügung stellen. Allein 37% der 674 Befragten

[42] vgl. DAFERMOS, George (2001) " Management and Virtual Decentralized Networks: The Linux Project ", elektronisch veröffentlicht unter http://opensource.mit.edu/papers/dafermoslinux.pdf [Stand: 26.09.2002], S. 52
[43] vgl. PAVLICEK S. 148f
[44] LAKHANI (2002), S. 11f

in der Studie der BCG/OSDN gaben an, dass sie an ihrem Arbeitsplatz direkt an Open Source Projekten mitwirken und somit bezahlte, vom Arbeitgeber bewilligte, Arbeitszeit nutzen. Über drei Viertel der somit subventionierten Entwickler werden ganz gezielt dahingehend unterstützt, da sie an Projekten mitarbeiten, die für den Arbeitgeber bzw. das Unternehmen von Bedeutung sind[45].

Privateigentum und Effizienz

In der Wirtschaftstheorie wird Eigentum eine zentrale Institution angesehen, unter der man "ein auf ein bestimmtes Zielbündel abgestelltes System von Normen einschließlich deren Garantieinstrumente"[46] versteht. Institutionen haben den Zweck, individuelles Verhalten zu steuern und Unsicherheiten zu vermindern. Man geht heute davon aus, dass der ökonomische Anreiz des Privateigentums im Allgemeinen zu "effizientem, nicht verschwenderischem Einsatz knapper Ressourcen"[47] beiträgt.. Der Softwareriese Microsoft beweist anhand seiner Umsatzzahlen, dass sich durch den restriktiven Umgang mit Verfügungsrechten und Privateigentum Profite maximieren lassen. In diesem Sinne trifft das Paradigma des effizienten Privatbesitzes also völlig zu. Wie jedoch lässt sich dies auf Freie Software anwenden?

Dem Umkehrschluss nach ist ein öffentliches Gut der Gefahr des verschwenderischen Gebrauchs ausgesetzt. Da Freie Software oft als Gemeineigentum[48] angesehen wird, muss sie auf diesen Vorwurf hin überprüft werden. Geldflüsse sind im Open Source Bereich naturgemäß äußerst gering sind und so muss sich ein

[45] vgl. LAKHANI (2002), S. 36
[46] MEFFERT, Heribert(2000): "Dienstleistungsmarketing", Wiesbaden (Gabler), S.63
[47] RICHTER (1999), S. 81
[48] vgl. Titel GRASSMUCK (2002)

anderer Maßstab für die Effizienz finden lassen. Wie weiter oben beschrieben, steigt der Wert einer Software durch positive Netzeffekte. Wenn sich nun nachweisen ließe, dass Projekte aus der Open Source Szene einen hohen Verbreitungsgrad genießen, kann die These gestützt werden, dass auch der Verzicht auf den restriktiven Gebrauch von Verfügungsrechten im Bereich der Freien Software eine wohlfahrtsfördernde und effiziente Wirkung haben kann.

Ein überzeugendes Beispiel für den hohen Verbreitungsgrad Freier Software liefert der Webserver Apache[49]. Ein Webserver ist ein Programm, das benötigt wird, um die Übermittlung von Webseiten realisieren zu können. Die größten Konkurrenten in diesem Markt sind Microsoft und die Apache Group. Während jedoch das kommerzielle Microsoft-Produkt "Internet Information Server" (IIS) laut einer Erhebung von Netcraft im August 2002 nur auf knapp 25% aller aktiven Webserver eingesetzt wird[50], steht der frei verfügbare Apache mit fast 67% unangefochten an der Spitze. Fügt man dieser Tatsache den Fakt hinzu, dass ein renommiertes Marktforschungs- und Beratungsunternehmen wie Gartner davon abrät, den Microsoft-Server einzusetzen, da die Fehleranfälligkeit des IIS zu "hohen Kosten"[51] führt, lässt sich auch ohne vorliegende Umsatzzahlen nachweisen, dass ein Open Source Projekt durchaus effizienter sein kann, als ein restriktiv geschütztes.

Organisation und Management

An Freien Software Projekten wird oft über die ganze Welt verteilt

[49] siehe auch http://www.apache.org
[50] "Netcraft Web Server Survey", elektronisch veröffentlicht unter http://www.netcraft.com/survey/ [Stand: 24.09.2002]
[51] "Gartner rät von Microsoft ab" elektronisch veröffentlicht unter http://www.heise.de/newsticker/data/ps-21.09.01-000/ [Stand: 24.09.2002]

gleichzeitig entwickelt. Da eine multilaterale Zusammenarbeit ein hohes Maß an Organisationstalent erfordert, sollen nun zwei der wichtigsten Medien der Organisation bzw. Kommunikation vorgestellt werden.

Mailinglisten

Die technisch einfachste und dennoch von den meisten Open Source Projekten genutzte Methode ist die Mailingliste. Bei einer Mailingliste handelt es sich um ein "automatisiertes eMail Verteilersystem, bei dem die Teilnehmerverwaltung mit An- und Abmeldung und die Verteilung der eMails an alle angemeldeten Teilnehmer automatisch"[52] erfolgt. Das heißt, das Verteilersystem sorgt dafür, dass eine Email von einem Mitglied an den Verteiler automatisch allen Mitgliedern per Email zugestellt wird.

Emails haben trotz ihrer technischen Einfachheit einen großen Einfluss auf Motivation und Innovationsgeist. Eine Reihe von Experimenten an der Carnegie-Mellon-Universität in Pittsburgh belegt, dass die Verwendung von Emails gegenüber direkten Gruppengesprächen, wie sie zum Unternehmensalltag gehören, zu "mehr praktischen Vorschlägen"[53] führte. Dies mag an folgenden Unterschieden von Emails zu Gesprächen liegen[54]:

1. Emails sind **asynchron**
2. Emails sind **computerlesbar**
3. Emails sind **ortsungebunden**
4. Emails sind **archivierbar**

[52] Stichwort "Mailingliste" in iLexikon, elektronisch veröffentlicht unter http://www.ilexikon.net/m/mailingliste.htm [Stand: 25.09.2002]
[53] FREY, Hartmut (1999) "Email: Revolution im Unternehmen", Neuwied (Luchterhand), S. 33
[54] vlg. FREY (1999) S.31ff

Durch die Asynchronität und Ortsunabhängigkeit ergibt sich für Nutzer der Vorteil, dass selbst entschieden werden kann, wann und wo Emails bearbeitet werden. Open Source Projekte sind oft über mehrere Zeitzonen geographisch verteilt[55] und fordern somit zeitliche und örtliche Unabhängigkeit von den Medien der Kommunikation.

Da Emails in digitaler Form vorliegen und wenig Speicherplatz benötigen, sind sie sehr leicht archivierbar. Auf den Webseiten großer Open Source Projekte wie Apache, OpenBSD und Linux finden sich Links zu Archiven der geführten Diskussionen, die sich in Regel auch nach Schlagwörtern durchsuchen lassen. Auf diese Weise wird eine Wissensdatenbank kontinuierlich erweitert, das auch im Nachhinein viele Fragen beantworten kann. Eine typische Aufforderung lautet daher: "Before posting a question on misc or any other mail list, please check the archives [...]. While it might be the first time you have encountered the problem or question, others on the mail lists may have seen the same question several times in the last week, and may not appreciate seeing it again"[56].

CVS

Ein besonderes Tool zur Organisation von Open Source Projekten stammt aus der Freien Software Szene selbst. Es handelt sich um das "Concurrent Version System"[57] (CVS), mit dem ein mächtiges Werkzeug zur Verfügung steht, dass es technisch ermöglicht, mehrere Entwickler an einem Projekt arbeiten zu lassen. CVS ist dabei dafür zuständig, die Rechte der Mitglieder an Dateien zu verwalten, zu speichern wer gerade an welcher Datei arbeitet und

[55] siehe auch "Weltkarte der [Debian] Entwickler", elektronisch veröffentlicht unter http://www.debian.org/devel/developers.loc [Stand: 24.09.2002] und Karte der OpenBSD Entwickler, elektronisch veröffentlicht unter http://www.openbsd.org/images/map.jpg [Stand: 24.09.2002]

[56] "Other OpenBSD information resources", elektronisch veröffentlicht unter http://www.openbsd.org/faq/faq2.html [Stand: 25.09.2002]

[57] vgl. http://www.cvshome.org/ [Stand: 27.09.2002]

Veränderungen im Nachhinein wieder rückgängig machen zu können. So kann durch ein CVS ein Projekt in jeden beliebigen Zustand wieder zurückversetzt und jede Änderung bzw. Erweiterung einem Teilnehmer zugeordnet werden. Es handelt sich hierbei um eine völlig neue Qualität von Medium, dessen umfassende Beschreibung hier den Rahmen sprengen würde. Es sei jedoch angemerkt, dass eine genaue Betrachtung dieser Technik an geeigneter Stelle sicher sehr aufschlussreich über den automatisiert verwalteten Arbeitsablauf in einer vernetzten Arbeitsumgebung sein kann.

Open Source Projekte als Organisationen

Um im Folgenden die Organisation, das heißt das Management, von Open Source Projekten untersuchen zu können, soll nun erklärt werden, warum Open Source Projekte selbst als Organisation aufgefasst werden können und worin aus ökonomischer Sicht der Grund ihres Entstehens zu sehen ist. Unter einer Organisation werden im Allgemeinen "gegliederte Gruppen von Personen in Verfolgung gemeinsamer Ziele"[58] verstanden, deren Ziel im Fall von Open Source Projekten darin besteht, eine funktionsfähige Software zu entwickeln bzw. zu erweitern.

Als entscheidender Grund für die Entstehung von Organisationen wird von der Ökonomie der Fakt angesehen, dass Informationen kein freies Gut sind[59]. Der Term "nicht frei" bezieht sich hier auf die Notwendigkeit, dass Informationen entdeckt, erfunden, gesucht, ausgehandelt oder getauscht werden müssen und nicht als öffentliches Gut unbegrenzt zur Verfügung stehen. Nach der Neuen Institutionenökonomik wird eine Organisation hierbei als Netzwerk

[58] RICHTER S. 292
[59] vgl. RICHTER S. 293

relationaler Verträge zwischen Einzelpersonen mit dem Zweck der Regelung ökonomischer Transaktionen zwischen einzelnen Angehörigen der Organisation aufgefasst[60]. Dabei fallen Transaktionskosten an, unter denen man Such-, Informations-, Verhandlungs-, Entscheidungs-, Überwachungs- und Durchsetzungskosten versteht[61].

Die Organisation Open Source Projekt ist somit eine Institution die den einzelnen Teilnehmern einen günstigeren Zugang zu Informationen in Form von Programmcode und Wissen ermöglichen kann. Für Einzelpersonen sind die Transaktionskosten der Informationssuche inakzeptabel hoch, da die begrenzte Rationalität des Individuums die Suchkosten in unbegrenzter Höhe steigen lassen kann. Erst ein Zusammenschluss mehrerer Individuen in Form einer Organisation kann bei größeren Projekten diese Kosten begrenzen. Durch das Prinzip der Reziprozität ergibt sich für die Teilnehmer ein vorteilhafter Tausch.

Die Organisation eines Open Source Projektes: Debian

Um die typische Organisation von Open Source Projekten exemplarisch zu veranschaulichen, wurde das Projekt "Debian GNU/Linux" ausgewählt. Bei Debian handelt es sich um ein freies Betriebssystem auf der Basis von Linux, das mit über 8000 Softwarepakete ausgeliefert wird[62]. Im Gegensatz zu bekannten Linux-Distributionen wie RedHat und SuSE[63] ist Debian kein kommerzielles Projekt. Eine finanzielle Unterstützung kann nur in Form von Spenden realisiert werden.

---

[60] vgl. RICHTER S. 293
[61] vgl. RICHTER S. 51ff
[62] "Über Debian", elektronisch veröffentlicht unter http://www.debian.org/intro/about [Stand: 25.09.2002]
[63] vgl. http://www.redhat.com und http://www.suse.de

Organisationsstruktur

Nach außen präsentiert sich Debian mit einer übersichtlich gegliederten Organisationsstruktur, die sowohl **administrative** als auch **produktionsbezogene** Gruppen aufweist[64]. So befinden sich in der Gruppe der Direktion die Untergruppen Leitung, Technisches Komitee und Schriftführung. Bezeichnend ist, dass für sämtliche Gruppen eigene Emailadressen angegeben werden, die einen **direkten Kontakt** ermöglichen. Gehören einer Gruppe mehrere Mitglieder an, verbirgt sich hinter der Emailadresse eine **Mailingliste**, so dass alle Mitglieder eine Anfrage gleichzeitig erhalten und in eine Diskussion einbauen können.

Die Gruppe Öffentlichkeitsarbeit untergliedert sich in Ansprechpartner für Presseanfragen, Veranstaltungen und Partner-Programme und kann sich von der Struktur her mit professionellen PR-Abteilungen durchaus messen.

Ein Konglomerat aus diversen Bereichen findet sich in der Gruppe "Unterstützung und Infrastruktur", in der unter anderem die Verwaltung der Mailinglisten, der Support für neue Mitglieder aber auch die **Administration eigener Ressourcen** eingegliedert ist. Dem Projekt werden regelmäßig von Firmen wie Sun Microsystems, Compaq und Alpha Hardwarekomponenten und Rechnersysteme zur Verfügung gestellt, um den Entwicklern den Zugang zu diesen Technologien zu ermöglichen[65]. Für die **Sponsoren** ergibt sich daraus der bereits weiter oben beschriebene Vorteil positiver Netzeffekte. Da Debian Linux gerade im professionellen Serverbereich erfolgreich ist, können die Hardwarehersteller ihre

[64] vgl. "Debian's Organisationsstruktur", elektronisch veröffentlicht unter http://www.debian.org/intro/organization [Stand: 25.09.2002]
[65] vgl. Übersicht der Rechnersysteme, elektronisch veröffentlicht unter http://db.debian.org/machines.cgi [Stand: 26.09.2002]

teuren Systeme durch eine Kompatibilität zu Debian besser vermarkten.

Pakete und Maintainer

Die eigentlichen Debian-Entwickler sind in der Gruppe der "Distribution" strukturiert. Es lässt sich beobachten, dass das Debian System, wie die meisten Open Source Projekte, **modularisiert** ist. Dies bedeutet, dass das Projekt in voneinander unabhängige Einzelbereiche untergliedert ist, wodurch erst die (weltweit) verteilte Bearbeitung einer solch Umfangreichen Programmsammlung möglich wird[66]. Für die Entwicklung hat dies den Vorteil, dass Veränderungen an einem Teil des Systems nicht das ganze System negativ beeinflussen können: "Modularity means that any changes done, can be implemented without risk of having a negative impact on any other part"[67].

Die über 8000 Software Pakete unterliegen der Verantwortung einzelner "**Maintainer**", die persönlich die Entwicklung des jeweiligen Paketes betreuen. Jedes Paket besitzt einen eindeutigen Namen und somit eine zugehörige Emailadresse, unter der der aktuelle Maintainer zu erreichen ist: "Users can address questions to individual package maintainers using email. To reach a maintainer of a package called xyz, send email to xyz@packages.debian.org"[68]. Für Außenstehende ergibt sich eine äußerst **transparente Organisation**, deren Aufbau fast mathematisch nachvollziehbar ist. Innerhalb der Organisation Debian ist die **Verteilung der Kompetenzen** für jeden ersichtlich.

[66] vgl. KOGUT, Bruce und METIU, Anca: "Distributed Knowledge and the Global Organization of Software Development", elektronisch veröffentlicht unter http://opensource.mit.edu/papers/kogut1.pdf [Stand: 26.09.2002]
[67] DAFERMOS (2001), S. 43
[68] "The Debian GNU/Linux FAQ", Kapitel 11, elektronisch veröffentlicht unter http://www.debian.org/doc/FAQ/ch-support.en.html [Stand: 27.09.2002]

Streitkultur

Wie oben bereits beschrieben, wird bei der Entwicklung von Open Source Projekten auf einen hohen Grad an Qualität Wert gelegt. Dies führt zu einer beabsichtigten Streitkultur, die als **kreativ** angesehen wird, aber auch im Zaum gehalten werden muss. Entwickler werden dazu angemahnt, **diskussionsfreudig** aber auch **kompromissbereit** zu sein: "A sound and vigorous debate is important to ensure that all the aspects of an issue are fully explored. When discussing technical questions with other developers you should be ready to be challenged. You should also be prepared to be convinced! There is no shame in seeing the merit of good arguments"[69]. Dieser Appell richtet sich an die Einsicht des Programmierers und nicht seine Nächstenliebe. Qualität steht in der Open Source Szene als Maßstab für Recht und Unrecht. Nicht der Klügere soll nachgiebig sein sondern der Einsichtige und sich nicht dafür schämen müssen, einen falschen Standpunkt vertreten zu haben.

Zur **Schlichtung von Konflikten** stehen im Debian Projekt zwei übergeordnete Instanzen zur Verfügung: Das mehrköpfige "**Technische Komitee**" kann bei technischen Diskussionen als unparteiischer Gutachter hinzugezogen werden oder aus eigenem Ermessen handeln und Entwickler zur Berücksichtigung von Hinweisen auffordern. Sämtliche weitergehende Konflikte wie zum Beispiel Fehlverhalten von Mitgliedern (in den Mailinglisten können die Diskussion teilweise doch sehr persönlich sein) werden dem "**Leader**", der zweiten übergeordneten Instanz, vorgetragen[70]. Es lassen sich hier Ansätze einer Gewaltenteilung erkennen, die sogar konstitutionell verankert ist.

---

[69] "Debian Technical Committee" (2002), elektronisch veröffentlicht unter http://www.debian.org/devel/tech-ctte [Stand: 27.09.2002]

[70] "Debian Technical Committee" (2002)

Konstitution

Zur Manifestierung der bereits gezeigten Strukturen und Regeln wurde eine eigene "**Debian Constitution**"[71] geschrieben, die neben dem "**Gesellschaftsvertrag**"[72] und den Software **Lizenzen** die rechtliche Grundlage des Debian Projekts bildet. Trotz der größtenteils "ehrenamtlichen" Mitarbeiter, die ohne schriftliche Verträge ihre Zeit zur Verfügung stellen, hat Debian somit auch eine vertragliche Grundlage, die als Institution den Teilnehmern einerseits Sicherheit gibt, ihnen andererseits aber auch bestimmte Verhaltensregeln abverlangt und somit das Risiko eines Erwartungsbruches reduziert. Sowohl Mitglieder der Organisation Debian als auch Außenstehende können eine Erwartungshaltung aufbauen, welche die Grundlage für eine Zusammenarbeit bildet.

Führt man sich vor Augen, dass es sich beim Debian um 5000 größtenteils "ehrenamtliche" Mitarbeiter handelt, erstaunt die konzeptionelle Umsetzung des Managements. Von Laienarbeit, wie kostenlose Arbeit oft angesehen wird[73], ist hier nichts zu spüren. Im Gegenteil – Debian zeigt, dass auch nichtkommerzielle Unternehmungen durchaus effizient und klar strukturiert sein können, ohne dass der Druck des Geldes den Teilnehmern eine bestimmte Verhaltensweise oktroyiert.

[71] "Debian Constitution v1.0" (2002), elektronisch veröffentlicht unter http://www.debian.org/devel/constitution [Stand: 27.09.2002]
[72] "Debian-Gesellschaftsvertrag" (2002), elektronisch veröffentlicht unter http://www.debian.org/social_contract [Stand: 27.09.2002]
[73] BADELT (2002), S. 574

Zusammenfassung

Open Source Projekte begleiten die Computerindustrie seit ihren Anfängen. Wurden sie in den siebziger Jahren noch von der Wirtschaft gefördert, sehen sie sich seit den Achtzigern mit einer wachsenden kommerziellen Softwarebranche konfrontiert. Dass Open Source als neues Phänomen der eingesessenen Softwarebranche Konkurrenz macht, ist somit nicht richtig.

Aus ökonomischer Sicht eröffnen Open Source Projekte neue Sichtweisen auf die Motivation für den Einsatz von Arbeitskraft. Dem bisherigen Paradigma, dass das Bedürfnis nach Liquidität der ultima ratio des Managements ist, steht eine ernstzunehmende Alternative gegenüber. Menschen brauchen eine Grundversorgung, das kann nicht abgestritten werden. Das kostenlose Programmieren steht dennoch nicht im Gegensatz zu wirtschaftlichen Thesen, welche den Individualismus als Motor der Gesellschaft sehen. Zwar wird immer wieder versucht, eine neue Cybergesellschaft zu propagieren, die die Wirtschaftsmodelle ad absurdum führen soll, wie diese Arbeit jedoch gezeigt hat, kann auch eine vermeintliche Kultur des Verschenkens aus egoistischen Motiven entstehen.

Die Existenz ehrenamtlicher Arbeit im kulturellen und sozialen Bereich überrascht heute niemanden mehr. Der Unterschied zu Freier Software ist nur der, dass digitale Güter kopiert werden können. Während also klassische ehrenamtliche Arbeit in der Regel nur einem begrenzten Personenkreis zugute kommt, kann Open Source durch seine Reproduzierbarkeit theoretisch unendlich oft genutzt werden.

Eine Untersuchung der Organisation Freier Software Projekte zeigt interessante neue Methoden des weltweit vernetzten Arbeitens auf.

Die Prozesse der Programmentwicklung und Qualitätssicherung sind einerseits sehr technokratisch organisiert, lassen aber andererseits fast organisch Diskussionen und Innovationen von außen zu. Die verwendeten Technologien wie Mailinglisten oder CVS haben hierbei eine zentrale Funktion. Fast autopoietisch schreibt sich das System der Open Source Entwicklung seine Struktur durch Reaktion auf eigene Entwicklungen selbst. Dieses Verhalten birgt sowohl für die Organisationstheorie als auch den Bereich der Medienkultur sicher noch ein interessantes Forschungsfeld ...