Nuss, Sabine (2002): *Zur Verwertung allgemeinen Wissens. Ein kapitalistisches Geschäftsmodell mit Freier Software*, in: Das Argument, Heft 248: Das Imperium des Hightech-Kapitalismus, 44. Jg, Nr. 5/6, S. 653-667

Sabine Nuss

Zur Verwertung allgemeinen Wissens.
Ein kapitalistisches Geschäftsmodell *mit Freier Software*

>Wissensgesellschaft<, >Informationsgesellschaft< oder >Informationszeitalter< sind Begriffe die eine neue Entwicklungsperiode moderner Gesellschaften kennzeichnen sollen. Sie bleiben Deckworte, die die gegenwärtige Ära ebenso wenig charakterisieren können, wie der Begriff >Dampfmaschinen-Gesellschaft< die Industrialisierung zu definieren vermocht hätte (Marcuse 2002). Sie zeigen allerdings an, dass jeweils ein bestimmter Aspekt in unterschiedlichen Perioden der kapitalistischen Produktionsweise bestimmend und neu für die Dynamik gesellschaftlicher Entwicklung ist. Doch am >Informationszeitalter< ist gerade nicht die >Information< das Neue, ebenso wenig wie in der >Wissensgesellschaft< das >Wissen<. Information und Wissen in dieser bis zur Inhaltslosigkeit getriebenen Abstraktion gab es immer. Was konkret neu ist, ist die *Art und Weise der Informationsverarbeitung, -weitergabe und -gewinnung:* >Die Transformation von Wissen geschieht nicht von selbst, sondern wird durch Arbeit, Informationsarbeit, erreicht. Informationsarbeit hämmert Keilschrift in den Stein, lässt Pergament- und Papierhandschriften entstehen, schafft Maschinen und Verfahren für den Druck mit bewegten Lettern, erstellt Inhaltsverzeichnisse, Register, Abstracts, Übersetzungen von Texten, produziert Datenbanken und schafft digitale Repräsentationen von multimedialen Objekten, die hypertextuell verknüpft elektronische Netzwelten erzeugen.< (Kuhlen 2002, 66). Mittels der elektronischen digitalen Ausdrucksform von Information und der Verbreitung und Vernetzung von persönlichen Computern und Zentralrechnern kann Wissen[1] heute anders generiert, verbreitet und vervielfältigt werden.

[1] Über die Abgrenzung und Definition von >Wissen< und >Information< gibt es selbst in der Fachwelt keine Einigkeit. Der Begriff des >Wissen< betont eher den sozialen Kontext der Entstehung und beinhaltet nicht messbare Anteile, während >Information< aus Wissen generiert wird, als solche in messbare Einheiten zerlegbar ist und damit ausgedrückt werden kann in Worten, Zahlen, in Bits und Bytes. In dieser Weise möchte ich die Begriffe hier begreifen. Für weitergehende Überlegungen zur Unterscheidung vgl. Kuhlen 2002.

Die Technologie der Digitalisierung in Kombination mit der globalen Vernetzung der informationsverarbeitenden Maschinen ist kurz gefasst als >Internet< in die Wahrnehmung der Öffentlichkeit gelangt. Sie greift *verändernd* in den kapitalistischen Verwertungsprozess ein, sowohl in die Sphäre der Warenzirkulation als auch der Produktion. Auf der Ebene der Warenzirkulation wird das kapitalistische Privateigentumsregime mit neuen Problemen konfrontiert. Durch die Informations- und Kommunikationstechnologien ist das Band zwischen Wissen und seinem Träger lockerer geworden. Solange es keine technischen Schranken gibt, können die digital gesendeten Informationen beliebig oft verbreitet werden, die Kosten dafür gehen gegen Null. Ein Lied von Madonna, eine Virenschutzsoftware, ein Text Adornos kann ohne Qualitätsverlust beliebig oft kopiert und verteilt werden, wenn es >frei< im Netz steht. In diesem Fall ist eine der Grundvoraussetzungen für den erfolgreichen Verkauf einer Ware potenziell gestört: die Knappheit von Gütern. Wenn Musik, Text, Software etc. zum freien Download im Netz zur Verfügung stehen, werden sie nur noch wenige kaufen. Um eine künstliche Knappheit für digitale Informationswaren herzustellen wird daher mittlerweile auf verschiedenen Ebenen versucht, den ungezügelten Datenfluss kontrollierbar zu machen. Techniken, sogenannte Kopier*schutz*technologien, werden entwickelt, die die herrschenden Copyright-Regeln auch im digitalen Warenverkehr funktional (für die Verwertung) durchsetzbar machen sollen. Diese Maßnahmen sind allerdings noch nicht wirklich erfolgreich. Kündigt beispielsweise eine Plattenfirma an, demnächst ihr neues Star-Album auf einer kopiergeschützten CD auf den Markt bringen zu wollen, dauert es nicht lange, bis genau diese Titel im Netz kursieren.[2] Bislang handelt es sich um ein Wettrennen zwischen Unternehme(r)n, die ihre Ware mit Verschlüsselungs- oder Kopierschutztechnologien schützen wollen, und >Crackern<[3], die diese Systeme regelmäßig zu brechen in der Lage sind. Wegen dieser Schwierigkeiten in der Durchsetzung des herrschenden Privateigentumsregimes auch für die digitale Welt werden technische Maßnahmen vom Gesetzgeber retrospektiv unterstützt bzw. flankiert. Beispiele für solches Vorgehen sind der DMCA (Digital Millennium Copyright Act) in den USA und die Europäische Urheberrechtsrichtlinie. Beiden gemein ist, dass sie die (technisch noch mögliche) Umgehung des Kopierschutzes unter Strafe stellen. Auf alle diese Maßnahmen

---

[2] So geschehen bei der amerikanischen Plattenfirma Farenheit Entertainment, die ihr neues Konzept zum Schutz von Urheberrechten der Öffentlichkeit vorgestellt hatte und mit der neuen CD des Country-Sängers Charley Pride praxisnah demonstrierten wollte. Dennoch fanden sich kurz darauf zuhauf MP3-Dateien des Künstlers im Internet. Url: http://www.disc4you.de/news/mai2001/051601_02.html.

[3] Zur Definition des Begriffs >Cracker< in Abgrenzung zu >Hacker< vgl. Gröndahl 2000.

einzugehen würde den Rahmen dieser Arbeit sprengen, worauf es mir ankommt, ist Folgendes: Der Umstand, dass Informationsprodukte mittels der neuen Technologien problemlos verbreitet und vervielfältigt werden können und die Maßnahmen, welche genau dies zugunsten privater und ausschließender Aneignung verhindern sollen, haben zu einer breiten gesellschaftlichen Auseinandersetzung um den Zugang zu Wissen im Allgemeinen geführt.[4]

Seit der Kapitalismus die dominierende Vergesellschaftungsweise ist, besteht eine permanente Spannung zwischen dem privaten Einschluss von Wissen oder Informationen und dem gesamtgesellschaftlichen Zugang dazu. Wissen und die Gewinnung von Information ist das Ergebnis von Kooperation und Interaktion von Gesellschaftsmitgliedern, es kann nicht von isolierten Individuen generiert werden (die gedankliche Konstruktion eines Individuums, das frei oder abstrahiert von Gesellschaft *sein* soll, ist an sich bereits eine ideologische Erfindung). Gesellschaftlichkeit als Entstehungsbedingung von Wissen konfligiert im Rahmen kapitalistischer Produktionsweise mit der Notwendigkeit des Einzelkapitals, Wissen exklusiv zurückzuhalten, um es verwerten zu können. Auf der mikroökonomischen Ebene findet dies seinen Ausdruck darin, dass der einzelne Kapitalist sein in den Waren vergegenständlichtes Wissen geheim halten möchte, um es vor Nachahmung und damit Mitkonkurrenten zu schützen. Makroökonomisch aber ist der ^Gesamtkapitalist^^ am möglichst kostensparenden Zugriff auf das gesellschaftlich erzeugte Wissen angewiesen. Dieser Widerspruch zwischen der Zirkulationsebene (in Waren vergegenständlichtes Wissen soll exklusiv sein) und der Produktionsebene (Wissen kann nur gesellschaftlich generiert werden) wird mittels rechtlicher Maßnahmen zum >Schutz< geistigen Eigentums, wie beispielsweise dem Patentsystem, zu kompensieren versucht. Aus dieser Logik heraus ist das Recht auf geistiges Eigentum, im engeren Sinne das Copyright, zeitlich limitiert. Für die erwähnte Spannung ist es charakteristisch, dass das Patentsystem selbst seit jeher ein umstrittenes Feld ist. Bürgerliche Ökonomen können sich nicht recht entscheiden, ob Patente unnötige Schranken für den Zugang zu Wissen darstellen und Innovationen daher eher hemmen oder ob sie notwendig sind, um Innovationen zu fördern, da Konkurrenten Forschungsergebnisse einfach nachahmen könnten ohne aber die Vorschuss-Investitionen getätigt zu haben. Diese Möglichkeit würde mit der Zeit den Anreiz, neue Innovationen zu entwickeln, ersticken (vgl. Gröndahl 2002). Neu an den digitalen Technologien ist also nicht

[4] Ausführlicher zu dieser Auseinandersetzung und insbesondere ihren

die Spannung zwischen Schließung und Öffnung von Wissen, sondern der Umstand, dass diese Spannung kulminiert bzw. eine neue Dimension gewinnt.

Dies lässt sich an Software besonders plastisch zeigen, da Sie originär vergegenständlichtes Wissen -- in Algorithmen ausgedrückt -- beinhaltet. Im Gegensatz dazu bestehen Musik-, Text- oder Bilddateien aus einer reinen Ansammlung von Daten ohne inhärente logische Rechenoperationen. Software ist eines der wichtigsten Informationsprodukte überhaupt, da es selbst das Rohmaterial für die gesamte Internet-Infrastruktur darstellt (dies ist immer und nur in Interaktion mit Hardware zu denken). Software, oder besser: die neuen Informationstechnologien (womit das Zusammenspiel mit Hardware besser adressiert ist) sind vielmehr entscheidendes Mittel der gegenwärtigen Epoche zur Veränderung der Produktionsprozesse. Informationstechnologie wird genutzt um Arbeit zu rationalisieren, um Zeit- und Raumbeziehungen zu beschleunigen, zu verkürzen, neu und anders zu organisieren (>Entgrenzung<), um bessere Kontrolle und tiefere Einblicke in Produktionsprozesse zu gewinnen, um Arbeitsprozesse lokal und global gleichzeitig zu zentralisieren und zu dezentralisieren.

Beim Verkauf der Ware Software stellen sich die eben dargelegten Probleme: Die einzelne Software-Firma ist daran interessiert, dass niemand ihr Produkt imitiert (kopiert) und verkauft oder verschenkt. Um das zu gewährleisten wird der Quellcode, die menschenlesbare Sprache des Programms, geheim gehalten (und Kopierschutztechnologien eingesetzt). Erklärt an einem Alltagsbeispiel wäre es das gleiche, wenn Großmutter unseren Lieblingskuchen zwar serviert, das Rezept aber geheim hält: Den Kuchen können wir zwar essen, bekommen aber nur anhand des Geschmacks oder der Beschaffenheit nicht heraus, welche Zutaten Großmutter verwendet hat. Proprietäre Software ist demnach von anderen Programmierern nicht einzusehen, d.h. Eingriffe und Veränderungen sind nicht möglich. Die Entwicklung bleibt beschränkt auf das einzelne Unternehmen, die Anzahl der Entwickler bleibt immer im Rahmen der gerade noch rentablen Kapazitäten. Andersherum gesprochen: Die potenziell mögliche Steigerung der Anzahl jener, die die Software weiterentwickeln und verbessern könnten, wird dadurch verhindert.

---

theoretischen Prämissen vgl. Nuss 2002.

Ich möchte im Folgenden zwei Unternehmen vorstellen, welche die Spannung zwischen Einschluss und Öffnung von Wissen mit Hilfe von quelloffener oder *Open Source* Software zu lösen versuchen. Dabei wird der Quellcode des Programms nicht zurückgehalten, sondern ist verfügbar (um beim Beispiel zu bleiben: Großmutters Kuchen wird also mit Rezept serviert und wir können ihn nun nachbacken oder aber auch variieren und mit vielleicht einer Prise Zimt verbessern). Interessant hierbei ist, wie ein Unternehmen mit in Software vergegenständlichten Wissen Geld verdienen kann, ohne dieses Wissen exklusiv und geheim zu halten. Das Open Source Geschäftsmodell scheint wegweisend für >Innovationen, Organisationsdesign und Leitungsfunktionen, die weit über Software hinausgehen< (Boston Consulting Group 2002). Um das besser verständlich zu machen, möchte ich einen kurzen Abriss über die Entstehung von Open Source Software geben.

#u#*Schließung und Öffnung von digitalisiertem Wissen: Eine kurze Geschichte über die Ursprünge von Open Source*

In den Anfängen der Computerindustrie in den sechziger Jahren gab es keinerlei Schwierigkeiten mit digitalisierten Informationen Geld zu verdienen, weil der Handel auf den Verkauf von Hardware und technischem Support beschränkt war. Software war lediglich ein Nebenprodukt und es war kein Problem, sie auszutauschen und gemeinsam daran zu arbeiten. Dies änderte sich mit dem Beginn ihrer Kommerzialisierung. Es entstand lizenzrechtlich abgesicherte, proprietäre Software, bei der der Quellcode zurückgehalten wurde. Richard Stallman, ein Programmierer, der in den siebziger Jahren am MIT beschäftigt war, wollte sich mit dieser Entwicklung nicht abfinden, beklagte das Ende der glorreichen Tage der Freiheit und der offenen Kooperation in der Software-Entwicklung. Er begann ein neues, eigenes und >freies< Betriebssystem unter dem Name GNU (rekursives Akronym für ***G****NU is* ***n****ot* ***U****nix*)[5] zu entwickeln. >Frei< hieß dabei lediglich, dass der Quellcode einsehbar ist (nicht etwa dass die Ware kostenlos angeboten werden muss). Zu Beginn der 90er Jahre, als fast alle Komponenten des GNU-Betriebssystems bis auf den Kernel (das >Herz< eines Betriebssystems) geschrieben waren, entwickelte ein Informatikstudent aus Helsinki, Linus Torvalds, unabhängig von Stallman und seinen Mitprogrammierern, *Linux*, einen >freien< Kernel. Beide kombiniert ergaben das Linux-basierte GNU-System, welches unter dem

[5] http://www.gnu.org/gnu/gnu-history.html

schlichten Namen *Linux* Berühmtheit erlangt hat. Seither wurde und wird explizit in diesem Geiste Software geschrieben, deren Quellcode frei zur Verfügung gestellt wird.

Verglichen mit geschlossener oder proprietärer Software werden quelloffener folgende Vorteile zugeschrieben: a) Sicherheit: Aufgrund des verfügbaren Codes können Programmierer oder Nutzer mit dem entsprechenden Know-how einsehen, wie das Programm funktioniert. D.h., die Kontrolle über den eigenen Computer ist gewährleistet. b) Flexibilität: Da der Code offen ist, kann er auf individuelle Zwecke und Bedürfnisse hin verändert werden. c) Qualität: Offener Code ist immer >work in progress<. Open Source Entwickler verbessern den Code ständig und geben die so weiter entwickelten Programmversionen nach Gesichtspunkten der Qualität frei und nicht nach Gesichtspunkten kommerzieller Verwertungszwänge. d) Niedrige Preise: Jeder kann den Quellcode von Open Source Software aus dem Netz laden. Nutzer, die nicht so vertraut mit dem Computer und dem Umgang mit seinen Anwendungsprogrammen sind, können Support, Dokumentation, Handbücher für das entsprechende Open Source Programm erhalten und bezahlen nur dafür, nicht aber für den Code. e) Schnelle und günstige Hilfe: Es gibt eine umfangreiche Gemeinde von Open Source Entwicklern (die >Community<), welche in Newsgroups und Mailinglisten organisiert sind und jedem, der Hilfe braucht oder Fragen hat zu helfen versucht. Kommerzieller proprietärer Software hingegen sagt man nach, dass der Support meist von schlechter Qualität und teuer ist.[6] f) Kooperation: Da die Teilnahme an einem Open Source Projekt in der Regel für jeden offen ist, können solche Projekte weltweit Talente anziehen, die andernfalls niemals hätten zusammengebracht werden können. Nutzer und Entwickler von Software verschmelzen in Personalunion und erhöhen damit die Feedback-Frequenzen.

1985 gründete Stallman die Free Software Foundation[7] mit dem Ziel, die Rechte der Software-Nutzer und -Entwickler zu stärken. Gemeint waren die Rechte ein Programm zu nutzen, zu studieren, zu kopieren, es verändern und verbreiten zu können.[8] Drei Jahre später

---

[6] >Den Preis für den besten technischen Support im Jahre 1997 von InfoWorld erhielt die Linux Nutzer-Gemeinde [...] Fast alle großen Software-Konzerne haben End-Nutzer Lizenzbestimmungen, die sie von jeglicher Haftung befreien, falls Probleme mit dem Programm Schwierigkeiten nach sich ziehen. [...] Großteils ist das kommerzielle Support-Personal auf Anfänger-Probleme ausgerichtet und ist kaum in der Lage, Fragen jenseits des Benutzer-Handbuchs zu beantworten.< (Niemi 1998)

[7] http://fsfeurope.org/documents/preamble.de.html.

[8] http://www.gnu.org/fsf/fsf.html

schuf er die General Public License (GPL), die diese Rechte auf Software verbriefen sollte und prägte den Begriff >Copyleft< als Gegensatz zu >Copyright<. Die GPL wurde eine wichtige Antriebskraft in der Freien Software-Entwicklung. Ihren Verfechtern nach zu urteilen war sie >ein Hack des Copyright-Systems, sie stellt das Copyright-Konzept auf den Kopf, schafft eine ganze Gemeinde, die weltweit kooperiert und ermöglicht die Entwicklung von Software von Menschen und für Menschen<.[9] Die Ablehnung privater Aneignung von Software-Codes war und ist für Stallman und seine Anhänger nicht nur eine Frage der größeren Effizienz von Software-Entwicklung, sondern auch ein Schritt in Richtung einer freieren Gesellschaft. In Abgrenzung zu Stallman formierte sich in den späten neunziger Jahren die Open Source Bewegung. Sie favorisierte zwar ebenfalls quelloffenen Code im Gegensatz zu proprietärem Code, lehnte aber jegliche Politisierung dieser Forderung ab. Stallman wurde als zu ideologisch kritisiert und das eigene Plädoyer für offenen Code mit rein pragmatischen Gründen legitimiert. Größere Effizienz und bessere Qualität sind die Argumente für Open Source, Kommerzialisierung ist ausdrücklich gewünscht. Neben der Freien Software-Bewegung, die hauptsächlich auf die Freiheit von Softwarecodes fokussiert, die aber den Verwertungszwang im Kapitalismus nicht weiter in Frage stellt, und der Open Source-Bewegung, die sich selbst als >rein pragmatisch< begreift, existiert nur eine Minderheit, die mit der spezifischen Produktionsweise von Freier Software ganz explizit das politische Anliegen einer Gesellschaft jenseits von Staat und Kapital verbindet. In ihrer Anschauung wird Freie Software als reine Gebrauchswertproduktion interpretiert, das heißt, gearbeitet und versorgt wird nach tatsächlichen Bedürfnissen. Damit entziehe sich Freie Software der kapitalistischen Verwertung und könne als >Keimform< einer künftigen nicht-kapitalistischen Gesellschaft Wirkung erzielen (Diskussionen dazu unter www.oekonux.de, eine kritische Auseinandersetzung damit: Nuss/Heinrich 2001).

Das, was aus der Bewegung der Freien Software als Open Source Bewegung hervorgegangen ist, hat mittlerweile einen breiten Konsens in der öffentlichen Debatte gefunden. Ein kapitalismus-kritisches Anliegen wird damit allerdings nicht länger verbunden. Im Gegenteil: Es wird betont, dass Open Source den freien Wettbewerb fördert.[10] Damit nehmen die Verfechter von Open Source natürlich auch eine politische Haltung ein, sie sind affirmativ gegenüber der herrschenden kapitalistischen Produktionsweise. Die Neugründung der Open Source Bewegung und ihre Abgrenzung von Stallman und seiner Freien Software-Ideologie

[9] http://www.free-soft.org/gpl_history

spiegelt sich in der Entwicklung der Lizenzpolitik wieder. Viele Lizenzen für die Nutzung von quelloffener Software wurden zwar nach dem Vorbild der GPL geschaffen, aber für die Bedürfnisse der Verwertung modifiziert. Die Organisation Opensource.org listet allein 23 verschiedene Open Source Lizenzen auf und GNU.org zählt um die 45 Lizenzen. Software-Lizenzen können grob in drei Kategorien aufgeteilt werden: die traditionelle *kommerzielle, geschlossene* Software, die *kommerzielle, quelloffene* Software und die *anti-kommerzielle, quelloffene* Software (letztlich ist das die *Freie* Software*).* Innerhalb der offenen Software wird also unterschieden zwischen >geschäftsfreundlich< und >weniger geschäftsfreundlich<.

Die GPL erlaubt das Kopieren, Verbreiten und Verändern von Software, verlangt aber zugleich, dass das Programm nur mit eben genau diesen Rechten wieder weitergegeben werden darf. >Copyleft meint, dass jeder, der solche Software verbreitet [...] die Freiheit, sie vervielfältigen und verändern zu dürfen, mit verbreiten muss< (www.gnu.org). In der Konsequenz heißt dies, dass aus quelloffener GPL-geschützter Software keine proprietäre generiert werden kann. Der GPL wird daher vorgeworfen, einen >Virus Effekt< zu haben, weil sie jeglichen Code, der mit ihr in Berührung kommt, mit >Freiheit< infiziert. Vor diesem Hintergrund ist einleuchtend, warum alternative Lizenzen für quelloffenen Code entstanden sind, die weniger kategorisch eben auch wieder die Schließung von Code oder die Kombination von proprietärem und offenem Code erlauben. Diese Formen sind >geschäftsfreundlich<.

#u#*Open Source und die Geschäftswelt*

1999 publizierte das populäre Zeitgeist-Magazin *Wired* eine Geschichte über Open Source. Die führenden Start-Ups in diesem Bereich wurden aufgelistet, angefangen bei >den letzten, die auf den Linux-Zug gesprungen sind, bis hin zu den frühen Adaptierern des kommerziellen Impulses< (Krueger in *Wired* 1999). Die Gewinne, die diese Unternehmen machen, kommen demnach >von überall<: Vom Bekleben der verpackten Software-Pakete mit Logos, von der Entwicklung von Hardware, die mit offener Software läuft und vom Verkauf des Supports. Es gibt demnach verschiedene Möglichkeiten Open Source Software als Geschäftsgrundlage zu nutzen. Auch die *big players* wie IBM sind u.a. aus wettbewerbspolitischen Gründen in das

[10] >Freiheit. Freiheit. Freiheit: Freie Software bringt Wettbewerb und Marktwirtschaft zurück in die Softwarebranche.< (aus einem Werbe-Flyer von www.linuxhotel.de).

Geschäft mit Linux eingestiegen um die Monopolstellung von Microsoft zu schwächen (vgl. Winzerling 2002).

Ein reines Dienstleistungsunternehmen steckt beispielsweise hinter dem Open Source Projekt Zope (Zope.org). Es liefert Software für umfassendes >Content Management<. Das Betreiber-Unternehmens hat 1998 entschieden, den Code an die >Community< frei zu geben. Von diesem Moment an wurde Zope populär und zog Tausende von Programmierern an, die seither weltweit an dem Programm mitarbeiten. Für Paul Everitt, Geschäftsführer der Firma, war das ein beeindruckender Schritt: >Wenn Du keine Leute findest, dann öffne deinen Code. Jeder wird bei dir mitarbeiten wollen. Es ist erstaunlich. Leute, talentierte Entwickler, lieben Open Source und deshalb wirst Du keine Einstellungsprobleme mehr haben, wenn du deinen Code öffnest.< (Sims 2000). Heute verdient das Unternehmen mit dem Support und Schulungen rund um die jetzt offene Software. Viele Organisationen und Institutionen interessieren sich mittlerweile für das Phänomen der Open Source Produktionsweise. Insbesondere Consultingfirmen und Unternehmen sind beeindruckt von der effizienten Art und Weise, in der Open Source entwickelt wird, von der hohen Arbeitsmotivation der Programmierer, von der Kreativität die dabei freigesetzt wird, von dem Anreiz, der von dieser Art der Entwicklung offensichtlich ausgeht. Die Boston Consulting Group (2002) hat eine Umfrage unter Open Source Programmierern durchgeführt, einen sogenannten >Hacker Survey<. Es wurde gefragt: >Wer sind die Hacker, die all diese großartige freie Software schaffen? Wieviel Zeit verwenden sie dafür? Warum tun sie das? Woher kommen sie<. Eine andere Studie betont zwei weitere Aspekte. Im Gegensatz zum frei zugänglichen Code von offener Software würde >geheimes und geistiges Eigentum zu einem Verhalten anreizen, welches ökonomische Aktivität offensichtlich weniger effizient sein lässt<; das Open Source Produktionsmodell ist >effizienter als hierarchische im-Haus-Modelle< (Kogut/Metiu 2001). Damit wird die herrschende Theorie, wonach nur privates Eigentum Anreize zu Arbeit generieren könne, auf den Kopf gestellt. Auf der Ebene der Warenzirkulation stellt sich die Frage, wie >kommerzielles Investment von einem öffentlichen Gut genannt Information Wert aneignen kann< (ebd.)? Mit Hilfe der folgenden Illustration eines Open Source Geschäftsmodells (anhand zweier Firmen) soll versucht werden, diesen Fragen näher zu kommen.

#u#*Zwei Beispiele: VA Software und Collab.net*

#u#VA Software

Das Unternehmen VA Software wurde 1993 gegründet, hat seinen Sitz in Fremont, Kalifornien, und beschäftigt derzeit 179 Menschen. Es entwickelte die populäre Entwicklungs-Plattform *SourceForge.Net*. Von überall her quer über den Globus können Entwickler dort ein Open Source Projekt anmelden, den entsprechenden Software-Code gemeinsam mit anderen entwickeln und ihr Vorgehen und ihre Arbeitsfortschritte über verschiedene Werkzeuge, wie beispielsweise Mailinglisten, Fehler-Verwaltungsprogramme, Versionskontroll-Instrumente usw. administrieren. *SourceForge.Net* ist damit eine Art Infrastruktur für eine kooperative und global über das Internet stattfindende Software-Entwicklung. Natürlich könnten die Programmierer eine solche Plattform auch selbst organisieren. Denn >jeder kann das<, wie Marc Merlin, Systemadministrator bei VA Software bestätigt.[11] Aber >es ist eine Qual. Programmierer wollen Code schreiben, sie wollen kein CVS (Concurrent Versions System[12]) ins Leben rufen, das macht keinen Spaß. Die Idee dahinter war also, das wir den Leuten, die einfach ihre Projekte machen wollen, ein solches System zur Verfügung stellen. [...] es wurde sehr erfolgreich<. Gegenwärtig wird *SourceForge.Net* von rund 460.000 Entwicklern weltweit genutzt. Insgesamt sind 44.528 Open Source Projekte registriert (Stand: 31. Juli 2002). VA Software hatte nie intendiert mit dieser Plattform Geld zu verdienen. Das Unternehmen entwickelte die Software in den 90er Jahren, zu einer Zeit, als das eigentliche Geschäft aus dem Verkauf von Linux-Computern bestand. Merlin: >VA wollte der Community etwas zurückgeben, weil wir Linux verkauft haben und das haben wir ja nicht geschrieben.< Darüber hinaus wollte VA auch die weitere Entwicklung von Open Source fördern. Mit der Zeit stellte sich heraus, dass das Hardware-Geschäft keine Gewinne einbrachte, besonders nach dem so genannten Dotcom-Crash 2000/2001. Konsequenterweise stieß VA das Hardware-Geschäft ab. Etwa zeitgleich kamen Anfragen von Firmen und anderen Interessenten nach der Software, mit welcher die Plattform *SourceForge.Net* selbst läuft. VA Software erkannte, dass die Internet Plattform -- ursprünglich der Open Source Community gratis zur Verfügung gestellt -- das

---

[11] Die Aussage entstammt einem Interview, das wir im Juni 2002 in Fremont führten. An dieser Stelle möchte ich mich ganz herzlich bei Markus Euskirchen bedanken, ohne dessen technische und mentale Unterstützung die Interviews nicht möglich gewesen wären.

[12] Ein Programm, das es Entwicklern ermöglicht, Änderungen im Quellcode zu administrieren und Informationen über die Art und die Gründe dieser Änderungen zu speichern und anderen bereitzustellen. Auf diese Weise entstehen verschiedene Versionen von Programmen, die anhand ihrer Versionsnummer identifiziert werden können.

Softwareprodukt sein könnte, mit dem sich Geld verdienen ließe. Offensichtlich gab es speziell in großen Konzernen, deren Büros quer über die Welt verteilt sind, Bedarf für eine Software, die globale und über das Internet koordinierte, dezentrale Software-Entwicklung ermöglicht. VA Software entschloss sich, das Geschäft vom Verkauf der Hardware auf Software und Dienstleistungen umzustellen. Bis zu diesem Zeitpunkt war der Code von *SourceForge* offen, im Zuge der Kommerzialisierung der Plattform wurde er geschlossen. Ein Ausschnitt des Interviews verdeutlicht die Beweggründe:

#z# Merlin: Ursprünglich gingen wir dazu über Support, Kundenanpassungen und Installationen zu verkaufen und hatten auch ein paar Kunden. *SourceForge.Net* hatten wir noch vollständig quelloffen. Seit Ende 2000 kam es dann häufiger vor, dass Firmen es schwer rechtfertigen konnten, hunderte oder tausende von Dollars für ein Produkt zu bezahlen, das offen ist. Sie brauchten zwar immer noch die Installation, die Kundenanpassung, sie brauchten wirklich das, was wir ihnen anboten. Aber sie konnten ihre Bosse nicht überzeugen, für etwas, das frei sein soll, einen Scheck zu unterzeichnen.
Frage: Das klingt, als wäre es eine rein mentale Angelegenheit?
Merlin: Fast ausschließlich. Sie brauchen das Gefühl, für ihr Geld etwas von Wert zu bekommen. Das ist etwas, was vielen kommerziellen Open Source Projekten passiert.
Frage: War das der Hauptgrund für euch, aus *SourceForge* geschlossene Software zu machen?
Merlin: So würde ich es sehen. Ich denke, die Hauptsache war, dass sogar unsere eigenen Direktoren gesagt haben, man könne mit einer Sache kein Geld machen, die *freie* Software genannt wird. Das ist natürlich diskussionswürdig. Red Hat[13] hat Erfolg damit.#zz#

Während des Transformationsprozesses von offener zu geschlossener Software nahm VA Software Kontakt zu all jenen Programmierern auf (>das waren nicht so viele<, Merlin), die an der Entwicklung von *SourceForge* mitgearbeitet hatten und bat sie um Erlaubnis für die Schließung des Projekts. Aus der Software *SourceForge* wurde eine kommerzielle Version entwickelt, die *SourceForge Enterprise Edition*. Dieses Software-Paket mit geschlossenem Code ist mittlerweile VA Softwares Vorzeige- und Haupt-Produkt. Zu den Kunden, die *SourgeForge* nutzen, gehören Unternehmen wie Pfizer, Hewlett Packard, Agilent Technologies und Organisationen wie die National Science Digital Library und die Open SystemC Initiative.

Ursprünglich angetreten um auf der Basis von Open Source Software Geld zu verdienen, ging VA nach einer Phase des Trial and Errors dazu über, das Potenzial der Open Source Produktionsweise in Unternehmen hinein zu tragen, also *das Produktionsmodell selbst* zu >verkaufen<. *SourceForge* gibt Unternehmen die Möglichkeit, den >Wissensfluss< innerhalb von global und dezentral agierenden Unternehmen zu zentralisieren, konzentrieren und kontrollieren. In den Worten von VA Software: >Entwickler sind produktiver. CSD (kollaborative Software Entwicklung) reißt interne Barrieren nieder, so dass Entwickler weniger Zeit damit verbringen müssen, ihre Arbeit zu dokumentieren und mehr Zeit haben, Code zu programmieren. [...] Manager können die Aktivitäten verfolgen. Mit der Möglichkeit, die Arbeit der Programmierer zu zentralisieren, können Manager leicht neue Möglichkeiten identifizieren, sie können Multi-Projektberichte erstellen und Probleme verhindern, bevor sie eintreten. Eine effektive CSD Plattform macht besser einsichtig, welche Erfahrungen Entwickler haben und das sorgt dafür, dass Manager einen besseren Zugang zu Experten rund um den Globus haben.< (VA Software 2002, Werbeflyer). Als weitere Vorzüge werden genannt: >kürzere Entwicklungszyklen, weniger Kosten und eine bessere Qualität von Software<. *SourceFourge* integriert demnach die Vorteile von Open Source in ein kommerzielles und proprietäres Produkt. VA Software bietet seinen Kunden aber nicht nur das technische Hilfsmittel für eine kooperative, raum-zeitlich fragmentierte Produktionsweise innerhalb des Unternehmens an, sondern unterstützt Firmen auch bei der Anwerbung freier Entwickler aus der weltweit verstreut lebenden Community für die Entwicklung spezifischer Open Source Software, auf deren Basis sie Geld verdienen wollen. Diese Strategie ist bei der zweiten Firma, die ich vorstellen möchte noch deutlicher ausgeprägt.

#u#CollabNet

CollabNet steht in direkter Konkurrenz zu VA Software. Das Unternehmen erzielt seinen Profit ebenfalls mit einer proprietären Entwicklungsplattform-Software namens *SourceCast* (gegenwärtig verkauft CollabNet die Software noch nicht als Produkt, sondern die Installations-, Wartungs- und Beratungs-Dienstleistungen darum herum). Brian Behlendorf, Mitgründer der Apache Software Foundation[14], gründete CollabNet im Sommer 1999. Die

---

[13] Red Hat ist ein internationales Software-Unternehmen und vertreibt Linux-Distributionen, das heißt, ein Betriebssystem, basierend auf Linux.

[14] Die als Open Source-Projekt entstandene Software Apache ist seit April 1996 die meistgenutzte Software für Webserver. Der *Netcraft Web Server*

Firma hat ihren Sitz in Brisbane, Kalifornien, und beschäftigt derzeit rund 85 Menschen. Neben ihrem Kerngeschäft, dem Vertrieb und der Vermarktung von *SourceCast* betreibt CollabNet ebenfalls eine öffentliche Internet-Entwicklungsplattform, mittels der Programmierer weltweit Open Source Projekte entwickeln können. Diese Internet-Seite unter dem Namen *Tigris.org* stellt das Pendant zu VA Softwares *SourceForge.Net* dar. Interessant hierbei ist, dass zwei fest angestellte Programmierer von CollabNet für je zwei Open Source Projekte bei *Tigris.org* verantwortlich zeichnen. Sie haben die Software-Projekte gegründet und betreuen sie weiter. Eines der beiden Projekte, *Subversion*, möchte eine neue Versionskontroll-Software entwickeln, da man mit der bestehenden (CVS) allgemein unzufrieden sei. Gegenwärtig hat dieses Projekt ein halbes Dutzend reguläre Entwickler und ungefähr ein Dutzend Programmierer mit >commit privileges<.[15] Diese Entwickler sind Teil der Open Source Community und nicht bei CollabNet angestellt. Abgesehen von diesen gibt es noch die vereinzelten und nicht regelmäßigen Beiträge in Form von Fehlermeldungen und kleinen Korrekturen, die von der weltweit verstreuten Open Source Community via Mailingliste geschickt werden. Diese ständigen Verbesserungen des Programms werden in das kommerzielle Produkt integriert, wobei hier die kommerzielle, geschlossene Software von CollabNet mit der offenen, von der Community entwickelten Software kombiniert wird, abgesichert mit einer entsprechenden Lizenz, wie oben erklärt. Ebenso verfährt CollabNet mit dem Open Source Projekt *Scarab.* Es bezweckt die Entwicklung einer Fehlerverwaltungssoftware und hat eine kleinere Beteiligung als *Subversion*, aber die Art und Weise, wie Wissen von der Community in das kommerzielle Produkt eingeht, ist die gleiche. Es erscheint ersteinmal unverständlich, warum ein Unternehmen für ein Programm wie *Scarab*, verpackt im Gesamtpaket *SourceCast*, überhaupt bezahlen soll, wenn es doch mit offenem Code frei verfügbar im Netz steht. Im Interview[16] erklärte Greg Stein, einer der Programmierer von CollabNet, dazu: >Die Leute in der Open Source Community haben eine gut funktionierende Version von Scarab. Aber Scarab als Teil von SourceCast funktioniert besser.< CollabNet betrachtet den Vorgang, *Scarab* so in ihr kommerzielles Produkt zu integrieren, dass es als Teil eines >Gesamtkunstwerks< funktioniert, als >value adding<.

Survey vom Mai 2002 zufolge laufen 56 Prozent aller Webseiten auf Apache (www.apache.org).

[15] Sogenannte >commit privileges< erhalten Mitarbeiter, die regelmäßige und brauchbare Beiträge zu einem Projekt liefern. Ein >Commiter< hat als quasi >verpflichteter Freiwilliger< Schreibrechte für den Quellcode und Stimmrechte bei Entscheidungen über die Zukunft eines Projektes.

[16] Interview vom 6. Juni 2002, Brisbane/Kalifornien. Neben Stein nahmen außerdem der Geschäftsführer und Chefentwickler von CollabNet Brian Behlendorf; Jon Stevens, ein weiterer CollabNet-Programmierer und Jason

CollabNet verkauft demnach nicht einzelne Programme, die in der Open Source Community entwickelt werden und dort verfügbar sind. Vielmehr stellen sie diese Programme nach spezifischen Kundenwünschen zu einem anwendungsfreundlichen und sinnvollen Ganzen zusammen und verkaufen sie als Paket, wobei geschlossener und offener Code in einem Programm-Paket nebeneinander existieren können. In den Worten von Brian Behlendorf: >Wir haben all diese verschiedenen Werkzeuge und alle sind als individuelle Werkzeuge irgendwie nützlich, d.h., die Leute können diese Programme auf ihren Computern selbst installieren und laufen lassen. Aber was schon immer schwierig war, ist diese Werkzeuge zu einem Gesamten zu integrieren, damit sie eine ^Suite^^ darstellen. Diese Integration braucht Zeit und Erfahrung. [...] Fast jede Firma draußen im Open Source Raum hat heutzutage dieses *value add*, aus dem sie dann einige Bits Software selbst behält.< Es wäre zu simpel zu behaupten, dass Firmen wie CollabNet oder VA Software die Community für ihre Profitinteressen ausbeuten würden. Vielmehr besteht eine wechselseitige Beziehung, bei welcher der kommerzielle und der nicht-kommerzielle Bereich voneinander profitieren. Die beiden Open Source Projekte *Subversion* und *Scarab* sind für CollabNet so etwas wie das Rohmaterial für ihre kommerziellen Produkte, wobei diese Open Source Software umso wertvoller ist, je mehr Programmierer und Nutzer sie gebrauchen, weil sie dann entsprechend viel Mitarbeit anzieht. Behlendorf dazu: >Wir möchten, dass jeder in der Welt *Subversion* benutzt ohne uns einen Pfennig dafür zu bezahlen, weil das den Markt für uns schafft, auf dem wir die anderen Bits *on the top* verkaufen können. Es ist ein Spiel, ein Risiko, es könnte sein, dass wir dieses Ding aufbauen und niemand nutzt es, weil es nicht gut genug ist.< CollabNet kann mittels seines Open Source Engagements Wissen von überall auf der Welt anziehen, ohne die Menschen auf der Basis von festen oder flexiblen Verträgen einstellen zu müssen: >Wir haben drei Vollzeit-Entwickler an diesem *Subversion*-Projekt sitzen, die seit 18 Monaten an dem Instrument arbeiten. Aber es ist ein Projekt, für das wir, um den gleichen Grad an Qualität und Verlässlichkeit zu erreichen, um die gleiche Arbeit zu leisten, die in der Open Source Welt erbracht wurde, wahrscheinlich um die zehn Leute hätten anstellen müssen, wenn wir es innerhalb unseres Hauses hätten aufbauen wollen. Für Subversion haben wir also drei Leute angestellt, aber tatsächlich sind es zehn Entwickler.< (Behlendorf). Nutzer und Entwickler der Community ziehen ihren Vorteil aus dem Open Source Engagement von CollabNet und VA Software, weil sie eine Plattform mitsamt der technischen Infrastruktur (Bandbreite, Speicherplatz) kostenlos zur Verfügung gestellt bekommen und weil der Code offen zum Download bereit steht, so dass ihn jeder für seine individuellen Bedürfnisse

---

Robbins, Wissenschaftler an der Universität von Kalifornien, Institut für

zuschneiden kann. CollabNet unterstützt gegenwärtig die Open Source Entwicklung mit ungefähr 50 Prozent ihres Zeitbudgets: >In der Hälfte der Zeit, in der sich unsere Leute hinsetzen und Code schreiben, schreiben sie Kram, der -- sobald er gecheckt ist -- in ein Open Source Projekt geht. Die anderen 50 Prozent behalten wir selbst.< (Behlendorf).

Dieses Geschäftsmodell für Software-Herstellung -- teils *in-house* mit eigenen Leuten, teils mit Hilfe der Community -- bietet CollabNet seinen Kunden an, es wird sozusagen als >Konzept< vertrieben. Das soll kurz am Beispiel von Sun Microsystems erläutert werden. Sun Microsystems entwickelt *StarOffice*, zu vergleichen mit dem etwas bekannteren *Microsoft Office,* ein Büro-Software-Paket bestehend aus u.a. Textverarbeitungs-, Tabellenkalkulation- und Präsentationsprogrammen. Im Oktober 2000 gab Sun den Quellcode für dieses Programm an die Community frei und benannte das damit gegründete Open Source Software-Projekt *OpenOffice.* Es existieren demnach zwei Office-Pakete: Die offene, kostenlose Version *OpenOffice,* (weiter)entwickelt von der Community und die kommerzielle Version *StarOffice*, entwickelt von Sun Microsystems. Die kommerzielle Version *StarOffice* kostet gegenwärtig um die 80 US-Dollar und richtet sich an Organisationen, Unternehmen und einzelne Konsumenten. *StarOffice* wird ausgeliefert mit Dokumentation und Dienstleistung, dazu gehören u.a.: Aktualisierungen des Programms auf CD-ROM, Installationshilfe, Nutzer-Dokumentation, rund um die Uhr Support via Internet, Garantieleistungen, Schulungen. *OpenOffice* hingegen richtet sich an Nutzer und Entwickler von freier Software, an die Open Source Community im allgemeinen. Grundsätzlich kann sich jeder, der zwar nicht unbedingt programmieren kann, aber ein wenig firm ist bei der Installation und Nutzung von Computerprogrammen, *OpenOffice* auf die Festplatte laden und damit arbeiten. Auch solche Nutzer tragen mit ihren Fehlermeldungen zur Verbesserung des Programms bei. Die Entwicklung von *OpenOffice* wird realisiert über die Infrastruktur, die CollabNet mit seiner Internet-Plattform *SourceCast* zur Verfügung stellt. Die Geschäftsstrategie ist auch hier: Die Verbesserungen und Entwicklungen, die die Community in *OpenOffice* einbringt, fließen in das kommerzielle Produkt *StarOffice* von Sun Microsystems. CollabNet dazu auf der Homepage: >Die Open Source Community benutzt nun Sun Microsystems‘ *OpenOffice.org* Internet-Seite um Suns *StarOffice Suite* zu verbessern [...] In weniger als zwei Monaten online hat diese Seite fast 1.700 Programmierer angezogen.< (www.collabnet.com).

CollabNet versucht also nicht nur bei seiner eigenen Software-Entwicklung von der kooperativen Wissensproduktion der Community zu profitieren, sondern >verkauft< dieses Konzept auch an die eigenen Kunden in Form von Dienstleistung (Beratung, Installation, Wartung und Hosting der Entwicklungsplattform *SourceCast*, Aufbau von Open Source Communities usw.). Die Software-Produkte der entsprechenden Firmen und auch von CollabNet selbst können dabei proprietär bleiben, entsprechende Open Source Lizenzen ermöglichen diese Vermischung von proprietärer und nicht-proprietärer Software (s.o.). Bei dieser Angewiesenheit der Unternehmen auf freie und unabhängige Entwickler könnte der Gedanke aufkommen, dass damit die Weiterentwicklung von Software gefährdet sei. Entwickler könnten von heute auf morgen ein Projekt >sterben< lassen, da sie in keinerlei Verpflichtungsverhältnissen stehen. Diese Befürchtung jedoch ist unbegründet, da Open Source Software Projekte quelloffen im Netz zur Verfügung stehen, so dass jederzeit Programmierer angestellt werden könnten, die am Programm weiter arbeiten -- wenn es nicht schon längst von anderen freien Entwicklern zur Weiterentwicklung in die Hand genommen wurde.

#u#Conclusio

CollabNet und VA Software sind Unternehmen, die einerseits das Open Source Modell unterstützen und es andererseits in die Unternehmenswelt implementieren, wobei die Verbindungsleistung zwischen der kommerziellen und der nicht-kommerziellen Sphäre Teil des Konzepts ist. Der Konflikt zwischen frei verfügbarem Code und der Notwendigkeit von privater Aneignung als Voraussetzung für Verwertung wird mittels entsprechender Open Source Lizenzen gelöst, so dass den Unternehmen der Zugang zum *Rohmaterial* offener Code erhalten bleibt, ohne dass sie selbst all ihren Code freigeben müssen. Die Lizenzen spielen demnach eine wichtige Rolle bei der in der Einleitung erläuterten Spannung zwischen Einschluss und Öffnung von Wissen, die dem Kapitalismus schon seit jeher inhärent ist. Die durch die Lizenzen verbrieften spezifischen >Property Rights< kompensieren den Widerspruch in einer Weise, dass der Zugriff des Kapitals auf das global verteilte Wissen für seine Zwecke konzentriert zur Verfügung steht. All die gutgemeinten Reden von >Freiheit<, all das Bemühen, die Freie Software auch als politischen Kampf in das Bewusstsein der Menschen zu rücken erzeugte -- und tut es noch -- bei vielen Menschen das Gefühl, es ginge hier um etwas essentiell Revolutionäres. Auch wenn es von Anhängern der Bewegung nicht

intendiert war, so ist die Öffnung des in Software vergegenständlichten Wissens auch für das Kapital eine Art Befreiungsschlag, der dazu führt, dass das Potenzial der Open Source Produktionsweise ausgeschöpft werden kann, ohne dass die Verwertungslogik des Kapitals davon gestört würde. Auch die viel gerühmte General Public License hat die Verhältnisse nicht wirklich auf den Kopf gestellt. Sie öffnete das Fenster für nicht-proprietäre Software, nutzte dabei aber explizit das Copyright um Copyleft durchzusetzen, bewegte sich also ganz kohärent im Rahmen des herrschenden Privateigentums-Regimes und stand so für Modifizierungen zugunsten der Kommerzialisierung zur Verfügung.

Die Anpassung der digitalen Welt an die herrschende Privateigentumsordnung wirkt in der Produktions- und in der Zirkulationssphäre auf je unterschiedliche Weise. Auf der Ebene der Zirkulation geht es darum, die Waren künstlich knapp zu halten, d.h., nur für das zahlungsfähige Bedürfnis zur Verfügung zu stellen. Dies verhält sich auch bei der Open Source Produktion nicht anders: Hier wird die Leistung von Unternehmen, Open Source zu einem anderen, neuen Produkt verarbeitet zu haben, knapp gehalten. Das Endprodukt selbst ist dann auch nicht mehr offen, sondern ganz oder in Teilen proprietär. Von Interesse ist wie bei Open Source die verschiedenen Lizenzen den Zugang der Unternehmen zum allgemeinen, offenen Wissen ermöglichen. Wie die Waren dann verkauft werden, nämlich doch wieder als exklusiv gehaltenes Wissen, ist keine neue Qualität des kapitalistischen Warentauschs.

Auf Ebene der Produktion verändert das Open Source Geschäftsmodell Arbeitsbedingungen in einer Weise, deren Auswirkungen und Beständigkeit gegenwärtig schwer abzusehen sind. Der wohl herausragendste Punkt ist die Integration von weltweit verstreuten, miteinander kooperierenden, unabhängigen und freiwilligen Entwicklern in den Arbeitsprozess kapitalistischer Unternehmen. Unternehmensberatungen, die sich damit beschäftigen, wie man das Letztmögliche aus Menschen herausholen kann, studieren schon seit einiger Zeit, welches Geheimnis sich hinter der hohen Arbeitsmotivation der Open Source Entwickler verbirgt, in der Hoffnung, daraus allgemein gültige Strategien ableiten zu können. Eines der Ergebnisse der bereits erwähnten Boston Consulting Group Umfrage ist, dass die Entwickler u.a. wegen >Ruhm und Ehre< an den Projekten mitarbeiten. Abgesehen von der persönlichen Anerkennung, die angestrebt wird, steht dieser Motivationsgrund aber auch für eine Art Profilierungsstrategie. (Noch unerfahrene) Programmierer können üben und sich unter Beweis stellen (auf der GNU.org Web-Seite sind die Programmierer, die viel zur Open Source Entwicklung beitragen, namentlich aufgelistet). Damit kommen die Entwickler auch den

Unternehmen näher, die im Netz jederzeit sehen können, welcher Programmierer sich gerade in welchem Projekt einen Namen macht.

Ein weiterer Grund für die Arbeitsmotivation liegt in der schlichten Tatsache, dass Programmierer das, was sie entwickeln, für ihre eigenen Zwecke brauchen. Dieser Zweck kann sowohl privater als auch geschäftlicher Natur sein -- falls das noch zu unterscheiden ist: Die Grenzen zwischen geschäftlichen und nicht-geschäftlichen Tätigkeiten scheinen im Bereich der Open Source Software Entwicklung zu verschwimmen. Keiner der Programmierer, mit denen wir sprachen, war in der Lage, seine Freizeit von seiner Arbeitszeit abzugrenzen. Aufgrund der freiwilligen und häufig unbezahlten Tätigkeit, die oft auch zu Hause, am Wochenende und/oder am Abend für die Entwicklung quelloffener Software geleistet wird und der Tatsache, dass aber ebenso häufig auch im Unternehmen während der Arbeitszeit Open Source entwickelt wird, ist die Trennung von Arbeit und Freizeit aufgehoben. Noch ein Wort zur politischen Motivation der Entwickler: Offensichtlich beruhigend war für das erwähnte Consulting Unternehmen der Umstand, dass für die Mehrheit der Programmierer die Bekämpfung proprietärer Software *nicht* der Grund für ihr Engagement ist.

Ob die Open Source Community tatsächlich Vorbildcharakter für künftige, neue Produktionsweisen haben wird, die weit über die Software-Industrie hinausreichen, bleibt abzuwarten. Aber es scheint nicht abwegig. Obwohl die Entwicklungsweise von Software ihre eigene Charakteristik hat und man das Open Source Modell nicht einfach auf andere Produktionsbereiche übertragen kann, können durchaus Impulse und Ideen von ihr ausgehen. Beispielsweise ist die Förderung der Loyalität der Entwickler eines Projekts von großer Bedeutung für die Motivation. Dazu wird fernab von selbstsüchtigem Eigennutz, der Gemeinsinn hochgehalten, das gemeinsame Engagement für ein gutes Produkt propagiert, der Gebrauchswert betont. Die Nützlichkeit und Qualität des Produkts kombiniert mit dem Gefühl, dies in Kooperation und *nicht gegeneinander* zu tun, sind Motivationsquellen, die neue Unternehmensstrategien, bzw. neue Arbeitsideologien, für sich nutzen könnten (im Gegensatz dazu wird bei der schon länger propagierten Teamarbeit häufig die Konkurrenz als motivationssteigerndes Element eingesetzt, indem man Teams gegeneinander konkurrieren lässt). Ein nächster naheliegender Gedanke wäre, das Potenzial der Freiwilligkeit stärker auszuschöpfen, zum Beispiel als künftige Einstellungsbedingung ein bestimmtes

>freiwilliges< Engagement vorauszusetzen. Damit wäre auch die Probe- und Einarbeitungszeit von den Unternehmen nach außen, in die >Eigenverantwortung< der arbeitenden Menschen verlagert. In einem anderen Punkt kann man von der Open Source Software Produktionsweise nicht viel neues lernen, da die Ökonomie schon vieles antizipiert hat -- dies betrifft die projektbezogene Tätigkeit der Programmierer. Projektbezogene Arbeit ohne Festanstellungsverträge steigern die Möglichkeit von Unternehmen, flexibel auf Markterfordernisse zu reagieren. Welche Lektion Open Source aber sicher liefert, ist, dass Arbeitsmotivation nicht hinreichend aus Privateigentum resultiert, wie es die herrschende bürgerliche Eigentumstheorie seit John Locke propagiert. Menschen arbeiten offensichtlich und mitunter gerne auch *um der Sache selbst willen*, frei von Zwang und Konkurrenz, wobei das Arbeitsergebnis dann allen zugute kommt. Ein Aspekt, den die Open Source Produktionsweise zwar nicht ^erfunden^^ hat, aber auf neue Weise wieder sichtbar gemacht hat -- hier wäre auch der theoretische und praktische Ansatz für eine wirklich emanzipative Alternative zur kapitalistischen Produktionsweise. Das heißt aber nicht, dass diese Produktionsweise nicht auch vom Kapitalismus selbst vereinnahmt werden kann, so, wie es gegenwärtig geschieht.

#u#Literatur

Boston Consulting Group/OSDN, 2002: *Hacker Survey*, in Cooperation with OSDN, Release 0.3, url: www.osdn.com/bcg/

Gröndahl, Boris, 2002: >Die Tragedy of the Anti-Commons. Kapitalistische Eigentumskritik im Patentwesen<, in: *Prokla* 126, 32. Jg., 89-101

ders., 2000: *Hacker*, Hamburg

Metiu, Anca, 2001: *Open source Software Development and Distributed Innovation*, url: http://jonescenter.wharton.upenn.edu/papers/2001.htm

Krueger, Patricia, 1999: >Tour de Source. A Guide to the Start-Ups<, url: www.wired.com/wired/archive/7.05/tour.html

Kuhlen, Rainer, 2002: >Napsterisierung und Venterisierung. Bausteine zu einer politischen Ökonomie des Wissens<, in: *Prokla* 126, 32. Jg., 57-88

Marcuse, Peter, 2002: >Entpolitisierte Globalisierungsdiskussion. Informationszeitalter und Netzwerkgesellschaft bei Manuel Castells<, in: *Prokla* 127, 32. Jg., 321-44

Niemi, David, 1998: *Open-Source Software: What is it, Why use it, And what's gotten into Netscape?*, url: www.tux.org/~niemi/opensource/customer-case.html

Nuss, Sabine, 2002: >Download ist Diebstahl? Eigentum in einer digitalen Welt<, in: *Prokla* 126, 32. Jg., 11-36

dies., und Michael Heinrich, 2001: >Warum Freie Software dem Kapitalismus nichts anhaben kann<, in: *Streifzüge* 1/2002, Wien, online: www.volkskunstschaffen.de/sabine_nuss/oekonux.htm

Sims, David, 2000: >Opening Zope: An Interview with Paul Everitt<, in: O'Reilly Network. Url: www.oreillynet.com/pub/a/network/2000/01/25/interview/

Winzerling, Werner, 2002: >Linux und die Freie Software. Eine Entmystifizierung<, in: *Prokla* 126, 32. Jg., 37-55