Stefan Meretz (April 2000)

»GNU/Linux ist nichts wert - und das ist gut so!«

Version 1.03, Letzte Änderung: 15.09.2000

Vortrag am 14.05.2000 bei den Braunschweiger Linux-Tagen und am 30.07.2000 beim Linuxtag 2000 in Stuttgart.

Ein Offline-Text basierend auf der Version 1.01 erschien auf der CD des Stuttgarter Linuxtages.

Originalquelle: http://www.kritische-informatik.de/lxwertl.htm

Ein open-theory-Projekt: http://www.opentheory.org/proj/linux-wertlos

Teil des Oekonux-Projektes.

Inhalt:

»GNU/Linux ist nichts wert - und das ist gut so!«

Man versteht wie etwas ist, wenn man versteht wie es geworden ist. Daher beginne ich mit einem kurzen Rückblick in die (Vor-) Geschichte Freier Software. Im zweiten Kapitel befasse ich mich mit der Frage, wie Freie Software ökonomisch in unser Wirtschaftssystem, den Kapitalismus, einzuordnen ist. Hieraus gewinne ich Kriterien für die Beleuchtung der scheinbar konträren Positionen von E. S. Raymond und R. M. Stallman, die stellvertretend für prominente Strömungen Freier Software stehen. Ich schließe ab mit einer Betrachtung der individuellen Handlungsmöglichkeiten und der Rolle, die Freie Software dabei spielen kann.

1. Eine kurze Geschichte Freier Software

Es gibt *freie* Software, weil es *unfreie* Software gibt. Unfreie Software ist »proprietäre Software«, also Software, die einem Eigentümer gehört [1]. Das wäre nicht weiter schlimm, würde die Tatsache des Privateigentums an Software nicht zum Ausschluß anderer führen. Der Eigentümer schließt andere von der Nutzung der Software aus, um ein knappes Gut zu erzeugen. Das geht bei Software relativ einfach durch Zurückhalten des Quellcodes des Programms. Nur knappe Güter besitzen Tauschwert und lassen sich zu Geld machen. Das ist das Funktionsprinzip des Kapitalismus [2]. Ich komme darauf zurück.

Unfreie wie freie Software gibt es noch nicht lange, gerade einmal ca. 20 Jahre. Die Entstehung unfreier wie freier Software versteht man, wenn man in Vorgeschichte schaut. Im Kalten Krieg, wir befinden uns in den 50er Jahren, wurde zwischen den USA und der Sowjetunion verbissen um die ökonomische Vorherrschaft gerungen. Vorherrschaft hatte damals eine militärische und eine symbolische Komponente, beide waren oft miteinander verwoben. So war es ein ungeheuerlicher Vorgang, als es der Sowjetunion 1957 gelang, den Sputnik in die Erdumlaufbahn zu schießen. Davon erholten sich die USA mental erst 1969, als sie es waren, die den ersten Menschen zum Mond brachten.

Der Sputnik wurde als technologische Niederlage erlebt. Sofort begannen hektische Aktivitäten, um den vermeintlichen Rückstand aufzuholen. 1958 wurde die ARPA (Advanced Research Projects Agency) gegründet, die die Aufgabe hatte, die Forschungsaktivitäten zu koordinieren und zu finanzieren. In einem Klima der Offenheit und Innovationsfreude wurden in der Folgezeit zahlreiche seinerzeit revolutionäre Produkte geschaffen, von denen ich zwei herausheben möchte, weil sie für die Freie Software eine besondere Bedeutung bekommen sollten: das Internet und das Betriebssystem UNIX (beide 1969). In diese Phase der staatlich finanzierten und koordinierten Forschung fällt auch die Festschreibung zahlreicher Standards, die heute noch Bestand haben [3].

Zum staatlichen Interesse an starken Standards kam das geringe Interesse der Computerindustrie an der Software. Computerindustrie war Hardwareindustrie, Software war Beiwerk zum Hardwareabsatz. Das änderte sich erst Ende der Siebziger Jahre als Computer immer leistungsfähiger wurden und Software auch eigenständig vermarktbar zu werden begann. In dem Maße, in dem Software zur profitablen Ware wurde, zog sich der Staat aus den Innovationen zurück. Um die je eigene Software verwerten zu können, mußte der Quelltext [4] dem Konkurrenten und damit auch dem User verborgen bleiben. Software war nur als proprietäre Software profitabel. Mit offenen Quellen hätte sich zum Beispiel Microsoft nie als monopolartiger Moloch etablieren können. Staatsrückzug und Privatisierung von Software bedeuteten jedoch auch eine Aufweichung von Standards. So entstanden in der Folge sehr viele zu einander wenig oder gar inkompatible Unix-Versionen (AT&T, BSD, Sun, HP, DEC, IBM, Siemens etc.).

Die Konsequenzen für den universitären Forschungsrahmen waren verheerend. Wo früher freier Austausch von Ideen herrschte, wurden jetzt Forschende und Lehrende gezwungen, Kooperationen zu beschränken oder ganz zu unterlassen. Software als Ergebnis von Forschungsaktivitäten durfte nicht mehr dokumentiert werden, sobald es über proprietäre Software an Firmen oder Patente gekoppelt bzw. selbst für die Patentierung vorgesehen war. Richard Stallman beschreibt diese Situation so:

> »1983 gab es auf einmal keine Möglichkeit mehr, ohne proprietäre Software einen sich auf dem aktuellen Stand der Technik befindenden Computer zu bekommen, ihn zum Laufen zu bringen und zu nutzen. Es gab zwar unterschiedliche Betriebssysteme, aber sie waren alle proprietär, was bedeutet, daß man eine Lizenz unterschreiben muß, keine Kopien mit anderen Nutzern austauschen darf und nicht erfahren kann, wie das System arbeitet. Das ist eine Gräben öffnende, schreckliche Situation, in der Individuen hilflos von einem ?Meister? abhängen, der alles kontrolliert, was mit der Software gemacht wird.« (Stallman 1999).

1.1. Der erste Geniestreich

Als Reaktion darauf gründete Stallman das GNU-Projekt [5]. Ziel des GNU-Projekts und der 1985 gegründeten Free Software Foundation (FSF) war die Entwicklung eines freien Betriebssystems. Hunderte Komponenten für ein freies Betriebssystem wurden entwickelt. Doch die wirklich geniale Leistung des GNU-Projekts bestand in der Schaffung einer besonderen Lizenz, der GNU General Public License (GPL) - auch »Copyleft« genannt. Die Lizenz beinhaltet auf folgende vier Prinzipien:

- das Recht zur freien Benutzung des Programms,
- das Recht, Kopien des Programms zu erstellen und zu verbreiten,
- das Recht, das Programm zu modifizieren,
- das Recht, modifizierte Versionen zu verteilen.

Diese Rechte werden gewährleistet, in dem die GNU GPL vorschreibt, daß

- der Quelltext frei jederzeit verfügbar sein und bleiben muß,
- die Lizenz eines GPL-Programms nicht geändert werden darf,
- ein GPL-Programm nicht Teil nicht-freier Software werden darf [6].

Die besondere Stärke der GNU GPL besteht in dem Verbot, GPL-Programme in proprietäre Software zu überführen. Auf diese Weise kann sich niemand offene Quelltexte aneignen und modifiziert in binärer Form in eigenen Produkte verwenden. Damit kann Freie Software nicht reprivatisiert werden, die Freiheit bleibt gewährleistet. Die besondere Stärke der GPL, die Reprivatisierung zu unterdrücken, ist in der Augen der Privatisierer ihr größter Nachteil. In der Folge entstanden daher zahlreiche Lizenzen (vgl. Tab. 1), die die strikten Regelungen der GPL aufweichten, um auch Freie Software kommerzialisierbar zu machen. Ich komme darauf zurück.

Lizenz-Eigen-schaften Soft-Ware-Art	Null-Preis	Freie Verteilung	Unbe-grenzter Gebrauch	Quellcode vor-handen	Quellcode modifi-zierbar	Alle Ab-leitungen müssen frei sein	Keine Ver-mischung mit pro-prietärer Software
Kommerziell (»Microsoft«)							
Probe-Software, Shareware	(X)	X					
Freeware (»Pegasus-Mail«)	X	X	X				
Lizenzfreie Libraries	X	X	X	X			
Freie Software (BSD, NPL, ...)	X	X	X	X	X		
Freie Software (LGPL)	X	X	X	X	X	X	
Freie Software (GPL)	X	X	X	X	X	X	X

Tab. 1: Vergleich der Lizenzarten

1.2. Der zweite Geniestreich

Das GNU-Projekt entwickelte ein nahezu komplettes Betriebssystem - bis auf einen kleinen, aber nicht unwichtigen Rest: den Kernel. Obwohl seit Beginn des GNU-Projekts geplant, gelang es nicht, einen GNU-Kernel zu entwickeln. Die mißliche Situation änderte sich 1991 schlagartig als Linus Torvalds die Version 0.01 eines freien Unix-Kernels vorstellte - fortan »Linux« genannt. Die Entwicklungsdynamik war rasant, der Erfolg war überwältigend - so überwältigend, daß heute oft vergessen und sprachlich verdrängt wird, welchen Anteil das GNU-Projekt am Zustandekommen des freien Betriebssystems hatte und hat.

Warum gelang aber einem finnischen Student, was einem ausgewachsenen Projekt wie GNU nicht glückte? Die Antwort ist nicht so naheliegend und einfach: Es lag am unterschiedlichen Entwicklungsmodell. Stallman und die GNU-Leute hatten die klassische Vorstellung, daß ein komplexes Programm wie ein Kernel nur von einem kleinen eingeschworenen Team entwickelt werden könne, da sonst der Überblick und die Kontrolle verloren gehen würde. Das hat Torvalds intuitiv auf den Kopf gestellt. Ein Ausschnitt aus der inzwischen in die Geschichte eingegangenen Tanenbaum-Torvalds-Debatte [7] verdeutlicht das. Tanenbaum schreibt:

> »Ich denke, daß die Koordination von 1000 Primadonnas, die überall auf der ganzen Erde leben, genauso einfach ist wie Katzen zu hüten ...
> Wenn Linus die Kontrolle über die offizielle Version behalten will und eine Gruppe fleißiger Biber in verschiedene Richtungen strebt, tritt das gleiche Problem auf.
> Wer sagt, daß eine Menge weit verstreuter Leute an einem komplizierten Stück Programmcode hacken können und dabei die totale Anarchie vermeiden, hat noch nie ein Softwareprojekt gemanagt.«

Torvalds antwortet:

> »Nur damit niemand seine Vermutung für die volle Wahrheit nimmt, hier meine Stellungnahme zu 'Kontrolle behalten' in 2 Worten (drei?):
> Habe ich nicht vor. [I won't]«

Torvalds veröffentlichte frühzeitig und in kurzen Zeitabständen neue Versionen. Es bildeten sich mehr und mehr freie Softwareprojekte, die ähnlich strukturiert waren und sind. Ältere Projekte strukturierten sich nach dem Vorbild von Linux um. Maintainer, einzelne Personen oder Gruppen, übernehmen die Verantwortung für die Koordination eines Projektes. Projektmitglieder steigen ein und wieder aus, entwickeln und debuggen Code und diskutieren die Entwicklungsrichtung. Es gibt keine Vorgaben wie etwas zu laufen hat, und folglich gibt es auch verschiedene Regeln und Vorgehensweisen in den freien Softwareprojekten. Dennoch finden alle selbstorganisiert *ihre* Form, die Form, die ihren selbst gesetzten Zielen angemessen ist. Das einfache Prinzip, das reguliert ist: Was funktioniert, das funktioniert! Ausgangspunkt sind die eigenen Bedürfnisse, Wünsche und Vorstellungen - das ist bedeutsam, wenn man freie und kommerzielle Softwareprojekte vergleicht.

1.3. Zusammenfassung zwischendurch

Überraschender weise besteht die historische Leistung von Richard Stallman und Linus Torvalds nicht in Softwarebausteinen, die sie entwickelt haben. Das haben sie auch getan, doch die eigentliche historisch-geniale Tat haben beide sozusagen »nebenbei« vollbracht. Stallman schuf die GNU GPL, die Lizenz, ohne die Freie Software undenkbar wäre. Es ist die Lizenz von Torvalds' Linux [8] und es ist die Lizenz, die dem Kapitalismus schwer im Magen liegt wie wir gleich sehen werden.

Torvalds hat intuitiv mit der alten hierarchischen Entwicklungsweise kommerzieller Software gebrochen. Ihm war die geldgetriebene Haltung des »ich muß die Kontrolle behalten« einfach zu blöd. Als pragmatischer Chaot hat er die Energien freigesetzt, von denen Freie Software lebt: die Selbstentfaltung des Einzelnen und die Selbstorganisation der Projekte.

2. Kapitalismus und Freie Software

Es gibt eine bekannte Comic-Vorstellung vom Kapitalismus. Oben gibt es die mit den schwarzen Zylindern, die über das Kapital und die Mittel zur Produktion verfügen. Unten gibt es die mit den blauen Overalls, die unter der Knute der Schwarzzylindrigen schwitzen, weil sie keine Produktionsmittel haben und deswegen ihre Arbeitskraft verkaufen müssen. Je nach persönlicher Vorliebe beklagt man, daß es ungerecht sei, daß die oben die unten ausbeuten, oder daß die Ausbeutung eben in der Natur des Unternehmertums liege.

Dieses Comic taugt nichts, schon gar nicht, wenn man »Freie Software« verstehen will. Ein anderes Bild muß her. Man kann den Kapitalismus als kybernetische Maschine verstehen, also einer Maschine, die »sich selbst« steuert. Das schließt ein, daß es keine Subjekte gibt, die »draußen« an den berühmten Hebeln der Macht sitzen, sondern daß die Maschine sich subjektlos *selbst* reguliert. Zentraler Regulator ist der (Tausch-)Wert [9], und zwar in zweifacher Weise: für die Seite der Produktion und die des Konsums.

2.1. Der Produktionskreislauf

In der Produktion wird *Arbeit* verrichtet. Sie heißt *konkret* insofern das Ergebnis ein Produkt ist, das auf ein Konsumbedürfnis trifft. Sie heißt *abstrakt*, weil es unerheblich ist, was produziert wird, Hauptsache es wird Wert geschaffen. Der Wert ist die Menge an Arbeitszeit, die in ein Produkt gesteckt wird. Werden auf dem Markt Produkte getauscht, dann werden diese Werte, also

Arbeitszeiten miteinander verglichen. Zwischen den direkten Produktentausch tritt in aller Regel das Geld, das keinen anderen Sinn besitzt, außer Wert darzustellen.

Was ist, wenn beim Tausch im einen Produkt weniger Arbeitszeit als im anderen steckt? Dann geht der Hersteller des »höherwertigen« Produkts auf Dauer Pleite, denn er erhält für sein Produkt nicht den »vollen Wert«, sondern weniger. Wer fünf Stunden gegen drei Stunden tauscht, verschenkt zwei. Das geht auf Dauer nicht gut, denn die Konstrukteure der Produkte, die Arbeiter und Angestellten, wollen für die volle Arbeitszeit bezahlt werden. Also muß der Tauschorganisator, der Kapitalist, zusehen, daß die für die Herstellung des Produkts notwendige Arbeitszeit sinkt. Das wird in aller Regel auf dem Wege der Rationalisierung vollzogen, dem Ersatz von lebendiger durch tote Arbeit (=Maschinen).

Was der eine kann, kann der Konkurrent auch. Wichtig und entscheidend ist dabei: Es hängt nicht vom Wollen der Konkurrenten ab, ob sie Produktwerte permanent senken, sondern es ist das Wertgesetz der kybernetischen Maschine, das sie exekutieren. Das Wertgesetz der Produktion besteht im Kern darin, aus Geld mehr Geld zu machen. Die Personen sind so unwichtig wie die Produkte, das Wertgesetz gibt den Takt an. Oder wie es der oberste Funktionär der Wertgesetz-Exekutoren, Hans-Olaf Henkel (BDI-Präsident), formuliert:

> »Herrscher über die neue Welt ist nicht ein Mensch, sondern der Markt. (...) Wer seine Gesetze nicht befolgt, wird vernichtet.« (Süddeutsche Zeitung, 30.05.1996)

2.2. Der Konsumkreislauf

Das Markt- oder Wertgesetz bestimmt auch die, nur ihre Arbeitskraft verkaufen können, um an das notwendige Geld zu kommen. Ohne Moos nix los. Auch die Arbeitskraft besitzt Wert, nämlich soviel wie für ihre Wiederherstellung erforderlich ist. Diese Wiederherstellung erfolgt zu großen Teilen über den Konsum, wofür Geld erforderlich ist, was wiederum den Verkauf der Arbeitskraft voraussetzt. Auch dieser Regelkreis hat sich verselbständigt, denn in unserer Gesellschaft gibt es kaum die Möglichkeit, außerhalb des Lohnarbeit-Konsum-Regelkreises zu existieren.

Beide Regelkreise, der Produktionskreis und Konsumkreis, greifen ineinander, sie bedingen einander. Es ist auch nicht mehr so selten, daß sie in einer Person vereint auftreten. Das universelle Schmiermittel und Ziel jeglichen Tuns ist das Geld. Noch einmal sei betont: Die Notwendigkeit, Geld zu erwerben zum Zwecke des Konsums oder aus Geld mehr Geld zu machen in der Konkurrenz, ist kein persönlicher Defekt oder eine Großtat, sondern nichts weiter als das individuelle Befolgen eines sachlichen Gesetzes, des Wertgesetzes. Eine wichtige Konsequenz dieser Entdeckung ist die Tatsache, daß unser gesellschaftliches Leben nicht von den Individuen nach sozialen Kriterien organisiert wird, sondern durch einen sachlichen, kybernetischen Regelkreis strukturiert ist. Das bedeutet nicht, daß die Menschen nicht nach individuellem Wollen handeln, aber sie tun dies objektiv nach den Vorgaben des kybernetischen Zusammenhangs. Wie Rädchen im Getriebe.

2.3. Knappheit und Wert

Damit die Wert-Maschine läuft, müssen die Güter knapp sein. Was alle haben oder bekommen können, kann man nicht zu Geld machen. Noch ist die Luft kein knappes Gut, aber schon wird über den Handel mit Emissionen nachgedacht, denn saubere Luft wird knapp. Viele selbstverständliche Dinge werden künstlich verknappt, um sie verwertbar zu machen. Das prominente Beispiel, das uns hier interessiert, ist die Software. Software als Produkt enthält Arbeit wie andere Produkte auch [10]. Wie wir im historischen Exkurs gesehen haben, war Software solange frei verfügbar wie sie nicht verwertbar erschien. Software wurde als Zugabe zur wesentlich wertvolleren Hardware verschenkt. Im Zuge gestiegener Leistungsfähigkeit und gesunkener Werthaltigkeit der Hardware (ablesbar an

gesunkenen Preisen) stieg auch die Bedeutung von Software - sie wurde auch für die Verwertung interessant.

Um Software verwertbar zu machen, muß Knappheit hergestellt werden. Dies geschieht im wesentlichen durch:

- Zurückhalten des Quellcodes
- Einschränkende Lizensierung und Patentierung

Wie bei jedem Produkt interessiert bei kommerzieller Software die Nützlichkeit und Brauchbarkeit den Hersteller überhaupt nicht. Ist ein aufgemotztes »Quick-And-Dirty-Operating-System« (QDOS) [11] verkaufbar, wird es verkauft. Ist das Produkt des Konkurrenten erfolgreicher, dann wird das eigene Produkt verbessert. Die Nützlichkeit und Brauchbarkeit ist damit nur ein Abfallprodukt - wie wir es zur Genüge von den kommerziellen Softwareprodukten kennen.

Entsprechend sieht es auf der Seite der Entwickler/innen aus. Auch Softwareentwickler/innen liefern nur ihre abstrakte Arbeit ab. In kaum einer anderen Branche gibt es so viele gescheiterte kommerzielle Projekte wie im Softwarebereich [12]. Mit 40 gehören Entwickler/innen schon zum alten Eisen. Der fröhliche Optimismus der Newbies im Business verfliegt schnell. Wer erlebt hat, wie gute Vorschläge mit dem Hinweis auf die Deadline des Projektes abgeschmettert wurden, weiß, was ich meine. Ein Berufstraum wird zum traumatischen Erlebnis.

2.4. Freie Software befreit

Das ist mit Freier Software anders. Der erste Antrieb Freier Software ist die Nützlichkeit. Der erste Konsument ist der Produzent. Es tritt kein Tausch und kein Geld dazwischen, es zählt nur eine Frage: Macht die Software das, was ich will. Da die Bedürfnisse der Menschen keine zufälligen sind, entstehen freie Softwareprojekte. Auch hier geht es nicht um Geld, sondern um das Produkt. Es gibt keine größere Antriebskraft als die individuelle Interessiertheit an meinem guten nützlichen Produkt und der individuellen Selbstentfaltung. Das weiß auch der Exekutor des Wertgesetzes in der Produktion. Deswegen spielt der Spaß, das Interesse am Produkt, auch in der geldgetriebenen Produktion eine wichtige Rolle. Es ist nur so, daß die abstrakte Arbeit immer vorgeht. Letztlich zählt eben nur, was hinten rauskommt - und zwar an Geld.

Abstrakte Arbeit ist nervtötend. Wer sagt, ihm mache seine abstrakte Arbeit Spaß, der lügt - oder macht sich was vor, um die abstrakte Arbeit aushalten zu können. Abstrakte Arbeit ist unproduktiver als freiwilliges Tun - wozu soll ich mich für etwas engagieren, was mich eigentlich nicht interessiert? Also muß man mich ködern mit Geld. Da sieht es für Informatiker/innen zur Zeit gut aus. Aber die Green Card bringt das auch wieder ins Lot. Dann ist da noch die latente Drohung: »Wenn du nicht gut arbeitest, setze ich dich woanders hin oder gleich ganz raus«. Wer sich bedroht fühlt, arbeitet nicht gern und schlecht. Zuckerbrot und Peitsche, die Methoden des alten Rom. Und Rom ist untergegangen.

Freiwilligkeit und nützliches Tun kann man nicht kaufen, jedenfalls nicht auf Dauer. Selbstentfaltung funktioniert nur außerhalb der rückgekoppelten Wert-Maschine. GNU/Linux konnte nur außerhalb der Verwertungszusammenhänge entstehen. Nur außerhalb des aus-Geld-mehr-Geld-machen-egal-wie konnte sich die Kraft der individuellen Selbstentfaltung zeigen.

2.5. Aber was ist mit den Geldmachern?

Machen wir uns keine Illusionen. Dort, wo man Geld machen kann, wird das Geld auch gemacht, und wenn es nicht anders geht, dann eben mit dem Drumherum von Freier Software. Das sind Absahner, nicht ohne Grund kommen sie alle zum Linuxtag. Das verurteile ich nicht, ich stelle es nüchtern fest. Maschinen haben den Vorteil, daß man ihre Wirkungsweise ziemlich nüchtern untersuchen kann. So sehe ich mir den Kapitalismus an.

Wenn ich die kapitalistische Wert-Maschine verstehe, habe ich nützliche Kriterien an der Hand - für das eigene Handeln und für die Einschätzung so mancher Erscheinungen Freier Software. Auf beides will ich im folgenden eingehen.

3. OSI und GNU - zwei verkrachte Geschwister

Anfang 1998 gründeten Eric. S. Raymond und Bruce Perens die Open Source Initiative (OSI). Erklärtes Ziel ist die Vermarktung von Freier Software, die Einbindung Freier Software in die normalen Verwertungszyklen von Software. Zu diesem Zweck wurde der Marketingbegriff »Open Source« ausgewählt. Nur mit einem neuen Begriff sei die Wirtschaft für die Freie Software gewinnbar. Der Begriff der »Freiheit« sei für die Wirtschaft problematisch, er klinge so nach »umsonst« und »kein Profit« [13]. Im übrigen wolle man das Gleiche wie die Anhänger der Freien Software, nur gehe man »pragmatischer« vor und lasse den »ideologischen Ballast« weg.

Richard Stallman, Gründer des GNU-Projekts, wirft den Open-Source-Promotern vor, in ihrem Pragmatismus würde die Grenzen zur proprietären Software verschwimmen. Der Begriff »Open Source« sei ein Türöffner zum Mißbrauch der Idee Freier Software durch Softwarefirmen, die eigentlich proprietäre Software herstellen und vertreiben. Im übrigen sei man überhaupt nicht gegen Kommerz und Profit, nur die »Freiheit« müsse gewahrt bleiben.

3.1. Der Wirtschaftsliberalismus von ESR

Nachdem sich OSI-Mitgründer Bruce Perens wegen der zu großen Anbiederung an den Kommerz wieder von der OSI abgewendet hat, ist es kein Fehler, sich alleine mit den Auffassungen von Eric. S. Raymond (ESR) zu beschäftigen. In den drei Aufsätzen »The Cathedral and the Bazaar«, »Homesteading the Noosphere« und »The Magic Cauldron« entwickelt er ein Kompatibilitätskonzept für die Verbindung von Freier Software und Kapitalismus [14].

Mit Freier Software ist der kommerzielle Software-Verwerter in eine Klemme gekommen. Freie Software ist öffentlich und nicht knapp. Die in der Freien Software steckende Arbeit wird einfach verschenkt. Damit ist sie nicht mehr verwertbar, sie ist wertlos. ESRs Bemühen dreht sich nun emsig darum, die aus den Verwertungskreisläufen herausgeschnittene Freie Software durch Kombination mit »unfreien Produkten« wieder in die Mühlen der kybernetischen Wert-Maschine zurückzuholen. Seine Vorschläge, die er in »The Magic Cauldron« entwirft - er nennt sie »Modelle für indirekten Warenwert« -, seien im folgenden kurz untersucht.

»Lockangebot«: Freie Software wird verschenkt, um mit ihr unfreie Software am Markt zu positionieren. Als Beispiel nennt ESR Netscape mit dem Mozilla-Projekt - ein Projekt, das mehrfach kurz vor dem Scheitern stand [15]. Was passiert hier? Eine Firma schmeißt ihren gescheiterten Browser den freien Entwickler/innen vor die Füße und ruft: »Rettet unsere Profite im Servermarkt!«. Dabei behält sie sich auch noch das »Recht« vor, die Ergebnisse wieder zu unfreier Software zu machen.

»Glasurmethode«: Unfreie Hardware (Peripherie, Erweiterungskarten, Komplettsysteme) wird mit einem Guß Freier Software überzogen, um die Hardware besser verkaufen zu können. Mußten vorher Hardwaretreiber, Konfigurationssoftware oder Betriebssysteme von der Hardwarefirma entwickelt werden, überläßt man das einfach der freien Software-Community. Wie praktisch, die kostet ja nichts! Unvergütete Aneignung von Arbeitsresultaten anderer - nennt man das nicht Diebstahl? Nein, werden die Diebe antworten, das Resultat ist doch frei!

»Restaurantmethode«: In Analogie zum Restaurant, das nur freie Rezepte verwendet, aber Speisen und Service verkauft, wird hier Freie Software von Distributoren zusammengestellt und zusammen mit Service verkauft. Die eigene Leistung besteht in der Zusammenstellung der Programme, der Schaffung von Installationsprogrammen und der Bereitstellung von Service. Unbezahlte Downloads oder gar Cloning der Eigenleistungen durch fremde Distributoren wird als Vergrößerung des gemeinsamen Marktes hingenommen. Oft werden gute Hacker von Distributoren angestellt, die Grenzen zwischen bezahlter und unbezahlter Arbeit sind fließend. Das Geschäftsgebaren der verschiedenen Distributionen ist durchaus unterschiedlich. Während sich das nichtkommerzielle Debian-Projekt mit ihrem Gesellschaftsvertrag [16] zur Einhaltung bestimmter Standards und zur Unterstützung Freier Software verpflichtet hat, steht für andere der reine Selbstzweck der Markteroberung im Vordergrund (etwa SuSE oder diverse Cloner).

»Zubehörmodell«: Hierzu gehören Herausgeber von Dokumentationen oder anderen Werken über Freier Software sowie andere Zubehörproduzenten, die nur auf der Welle mitschwimmen (etwa die Hersteller der Plüsch-Pinguine). Problematisch sind die exklusiven Lizenzen (Copyright), die eine Verbreitung schriftlicher Werke verhindern. Der Linuxtag ist selbst Opfer dieser Exklusion der Öffentlichkeit geworden. Verlage, die Texte vom letzten Linuxtag herausgebracht haben, sorgten dafür, das genau diese Texte von der Linuxtag-Website genommen werden mußten. Nur knappe Produkte eignen sich als Ware!

»Marketingmodelle«: Unter Ausnutzung der Popularität Freier Software werden verschiedene Marketingtricks aufgelegt, um proprietärer Software ein besseres Image zu verleihen und für Verkaufbarkeit zu sorgen. Damit sind noch nicht einmal die Betrüger gemeint, die sich einfach selbst das Label »Open Source« oder »Freie Software« auf ihre proprietären Produkte kleben, sondern Formen wie

- das Versprechen, proprietäre Software in Zukunft freizugeben;
- der Verkauf von Gütesiegeln, die erworben werden müssen um »Freie Software« verkaufen zu dürfen;
- der Verkauf von Inhalten, die eng mit dem sehr speziellen freien Softwareprodukt verbunden sind (etwa Börsenticker-Software)

Es sollte deutlich geworden sein, daß alle diese »Modelle für indirekten Warenwert« dazu dienen, die aus der Marktwirtschaft herausgefallene Sphäre Freier Software wieder zurück in den Kreislauf der selbstgenügsamen Wert-Maschine zu holen. Da kapitalistische Verwertung auf Knappheit und Ausschluß von Öffentlichkeit beruht, Freie Software aber genau das Gegenteil darstellt, müssen hier Feuer und Wasser in eine »friedliche Koexistenz« gezwungen werden. Doch wie es sich mit Feuer und Wasser verhält, so auch mit Freier Software und Verwertung: Nur eine kann sich durchsetzen.

Im neoliberalen Modell Freier Software von ESR gibt es folgerichtig keine wesentlichen Unterschiede zwischen »freier« Softwarelizenzen. Vermutlich hat er nur mit Magengrimmen die GPL trotz des enthaltenen Privatisierungsverbots auf die Liste von »OSI-zertifizierten« Lizenzen gesetzt, da man an der GPL nicht gut vorbeikommt. Bis auf die »Restaurantmethode«, dem Vertrieb Freier Software durch Distributionen, ist keines der oben genannten mit Buchstaben und Geist der GPL vereinbar [17].

Die GPL schließt künstliche Verknappung und Privatisierung von Code aus, und das behindert die Verwertung von Software weitgehend.

3.2. Der Bürgerrechtsliberalismus von RMS

Dem ökonomischen Liberalismus von ESR steht der Bürgerrechtsliberalismus von RMS entgegen. RMS argumentiert (1994), daß Software in privatem Besitz zu Entwicklungen führen würde, die dem gesellschaftlichen Bedarf entgegen läuft. Die Gesellschaft brauche

- Information, z.B. im Quellcode einseh- und änderbare Programme statt Blackbox-Software;
- Freiheit statt Abhängigkeit vom Softwarebesitzer;
- Kooperation zwischen den Bürgern, was die Denunziation von Nachbarschaftshilfe als »Softwarepiraterie« durch die Softwarebesitzer unterminieren würde.

Das seien die Gründe, warum Freie Software eine Frage der »Freiheit« und nicht des »Preises« sei. Bekannt geworden ist der Satz »Think of ’free speech’, not of ’free beer’«.

An diesen Kriterien orientiert sich auch die GNU GPL. Sie stellt sicher, daß Software dauerhaft frei bleibt oder ökonomisch formuliert: Sie entzieht Software dauerhaft der Verwertung. RMS ist dennoch keinesfalls gegen den Verkauf Freier Software (1996). Auch die GPL selbst ermöglicht ausdrücklich das Erheben einer Gebühr für den Vertrieb Freier Software.

RMS formuliert seine Vision gesellschaftlichen Zusammenlebens im GNU-Manifest von 1984 so:

> »Auf lange Sicht ist das Freigeben von Programmen ein Schritt in Richtung einer Welt ohne Mangel, in der niemand hart arbeiten muß, um sein Leben zu bestreiten. Die Menschen werden frei sein, sich Aktivitäten zu widmen, die Freude machen, zum Beispiel Programmieren, nachdem sie zehn Stunden pro Woche mit notwendigen Aufgaben wie Verwaltung, Familienberatung, Reparatur von Robotern und der Beobachtung von Asteroiden verbracht haben. Es wird keine Notwendigkeit geben, von Programmierung zu leben.« (Stallman 1984).

Eine schöne Vision, die ich ohne zu zögern teilen kann. Nur: Wer glaubt, diese Vision unter den Bedingungen der kybernetischen Verwertungsmaschine mit Namen Kapitalismus erreichen zu können, rennt einer Illusion hinterher. Der einzige Zweck der Wert-Maschine ist, aus Geld mehr Geld zu machen - egal wie, egal womit. Freiheit von Mangel, Muße, Freude, Hacking-for-Fun ist darin nicht vorgesehen.

Die von ESR mit angestoßene Open-Source-Welle führt das lehrbuchartig vor. Es geht überhaupt nicht mehr um gesellschaftliche Freiheit, die nur die Freiheit aller sein kann, sondern es geht um die Frage, wie ich aus etwas »Wertlosem« trotzdem Geld machen kann, wie ich die Freude der Hacker zu Geld machen kann, wie ich die Selbstentfaltung der Menschen in abstrakte, tote Arbeit verwandeln kann. Dieser mächtigen Welle vermag RMS mit dem Ruf nach »Freiheit geht vor« kaum etwas entgegenzusetzen. Vermutlich würde ESR antworten: Natürlich geht Freiheit vor, die ökonomische Freiheit!

Hieran wird deutlich, daß der Liberalismus eben zwei Seiten hat: Wirtschaftsliberalismus und Bürgerrechtsliberalismus. Robert Kurz arbeitet in seinem eindrucksvollen Werk »Schwarzbuch des Kapitalismus« (1999) die gemeinsame Verwurzelung im historischen Liberalismus heraus [18]. Er zeigt, daß auch der Bürgerrechtsliberalismus nur dazu da ist, Menschenfutter für die kybernetische Verwertungsmaschine zu liefern. Wer vom Kapitalismus nicht reden will, soll über die Freiheit schweigen.

4. Freie Software für freie Menschen

Wir sollten in die Offensive gehen! Wir sollten uns zum antikapitalistischen Gehalt der GPL bekennen! Wir können sagen »GNU/Linux ist nicht wert - und das ist gut so!«. Freiheit gibt es nur außerhalb der Verwertungs-Maschine. Die Freie Software da herausgeholt zu haben, war eine historische Tat. Jetzt geht es darum, sie draußen zu behalten, und nach und nach weitere Bereiche der kybernetischen Maschine abzutrotzen. Dafür gibt es zahlreiche Ansätze, die Stefan Merten im Beitrag »Gnu/Linux - Meilenstein auf dem Weg in die GPL-Gesellschaft« skizziert.

Wie kann das gehen, wird sich sicher so mancher fragen. Man kann doch nicht einfach rausgehen aus den Verwertungszusammenhängen - wovon soll ich leben? Das sind berechtigte, zwingende Fragen. Ich denke, daß es nicht darum geht, sofort und zu 100% aus jeglicher Verwertung auszusteigen. Es geht darum, einen klaren Blick für die Zwangsmechanismen der kybernetischen Verwertungsmaschinerie zu bekommen, und danach das individuelle Handeln zu bemessen. Ich will einige Bespiele nennen.

Konkrete und abstrakte Arbeit: Wenn ich für meine Reproduktion meine Arbeitskraft verkaufen muß, dann sollte ich nicht versuchen, darin Erfüllung zu finden. Natürlich ist es schön, wenn die Arbeit mal Spaß macht. Doch Lohnarbeit bedeutet abstrakte Arbeit, und dabei kommt es eben nicht auf meine Bedürfnisse, sondern die externen Zielvorgaben an. Selbstentfaltung gibt es nur außerhalb, z.B. in Freien Softwareprojekten. Wenn ich die Erwartungshaltung an die Lohnarbeit nicht habe, kann ich sie auch leichter begrenzen. Und das ist aufgrund des endlosen Drucks in Softwareprojekten eine dringende Notwendigkeit.

Eine Firma gründen: Manche denken, sie könnten der abhängigen entfremdeten Arbeit dadurch entkommen, indem sie eine eigene Firma gründen. Das ist so ziemlich die größte Illusion, die man sich machen kann. Als Firmeninhaber bin ich direkt mit den Wertgesetzen der kybernetischen Maschine konfrontiert. Die eigene Entscheidung besteht nur darin, in welcher Weise ich diese Gesetze exekutiere, welches Marktsegment ich besetze, welchen Konkurrenten ich aus dem Feld steche usw. Ich bin mit Haut und Haaren drin, muß permanent mein Handeln als das Richtige gegenüber allen rechtfertigen. Eine Distanzierung ist hier noch schwerer als bei der entfremdeten Lohnarbeit.

Verwertete Entfaltung: Die eigene Selbstentfaltung ist die letzte unausgeschöpfte Ressource der Produktivkraftentwicklung (Meretz 1999c). Das wissen auch die Exekutoren des Wertgesetzes, die die Selbstentfaltung der Verwertung unterordnen wollen. Sie bauen die Hierarchien ab, geben uns mehr Entscheidungsbefugnisse und Flexibilität bei der Arbeitszeit. Die Stechuhren werden abgeschafft, weil man sie nicht mehr braucht - alle arbeiten freiwillig länger nach dem Motto: »Tut was ihr wollt, Hauptsache ihr seid profitabel«. Die Zusammenführung der beiden Rollen des Arbeitskraftverkäufers und des Wert-Gesetz-Exekutors in einer Person ist der (nicht mehr so) neue Trick. Fallt darauf nicht rein! Die »Neue Selbständigkeit« kann zur Hölle werden [19], denn Verwertung und Selbstentfaltung sind unvereinbar.

Selbstentfaltung: Die unbeschränkte Entfaltung der eigenen Individualität, genau das zu tun, was ich wirklich tun will, ist nur außerhalb der Verwertungs-Maschine möglich. Nicht zufällig war es der informatische Bereich, in dem wertfreie Güter geschaffen wurden. Uns fällt es noch relativ leicht, das eigene Leben abzusichern. Wir werden gut bezahlt, finden schnell einen Job. Freie Software zu entwickeln, ist kein Muß, es ist ein Bedürfnis. Wir sind an Kooperation interessiert, und nicht an Verdrängung. Die Entwicklung Freier Software ist ein Beispiel für einen selbstorganisierten Raum jenseits der Verwertungsmaßstäbe. Nur dort ist Selbstentfaltung möglich.

Mit diesen Beispielen möchte ich für Nüchternheit, Klarheit und Offenheit plädieren - im Umgang mit anderen und sich selbst. Dazu gehört für mich auch, wieder über das gesellschaftliche Ganze zu sprechen, denn das sollten wir nicht den wirtschafts- oder bürgerrechtsliberalen Interpreten überlassen. Der Kapitalismus ist nichts dämonisches, man kann ihn verstehen und sein Handeln daran ausrichten. Dann hat Freie Software als *wertfreie Software* auch ein Chance.

5. Meta-Text

5.1. Versionen-Geschichte

- Version 1.00, 4.4.00: Erste Vorab-Version in der Oekonux-Mailingliste
- Version 1.01, 9.4.00: Version für die Linuxtage-CDs
- Version 1.02, 2.5.00: Aufgrund der open-theory-Diskussion überarbeitete Version
- Version 1.03, 15.9.00: Nochmalige Überarbeitung, Vorbereitung für die Übersetzung

5.2. Literatur:

DiBona, C., Ockman, S., Stone, M. (1999), Open Sources: Voices from the Open Source Revolution, Sebastopol/CA: O?Reilly; online verfügbar unter http://www.oreilly.com/catalog/opensources/book/toc.html.

Fischbach, R. (1999), Frei und/oder offen? From Pentagon Source to Open Source and beyond, in: FIFF-Kommunikation 3/99, S. 21-26.

Glißmann, W. (1999), Die neue Selbständigkeit in der Arbeit und Mechanismen sozialer Ausgrenzung, in: Herkommer, S. (Hrsg., 1999), Soziale Ausgrenzungen. Gesichter des neuen Kapitalismus, Hamburg: VSA

Kurz, R. (1999), Schwarzbuch Kapitalismus: Ein Abgesang auf die Marktwirtschaft, Frankfurt/Main: Eichborn.

Lohoff, E. (1998), Zur Dialektik von Mangel und Überfluß, in: Krisis, Beiträge zur Kritik der Warengesellschaft 21/22, Bad Honnef: Horlemann.

Meretz, S. (1999a), Die doppelte algorithmische Revolution des Kapitalismus - oder: Von der Anarchie des Marktes zur selbstgeplanten Wirtschaft. Internet: http://www.kritische-informatik.de/algorev.htm.

Meretz, S. (1999b), Linux - Software-Guerilla oder mehr? Die Linux-Story als Beispiel für eine gesellschaftliche Alternative. In: FIFF-Kommunikation 3/99, S. 12-21. Internet: http://www.kritische-informatik.de/linuxsw.htm.

Meretz, S. (1999c), Produktivkraftentwicklung und Subjektivität. Vom eindimensionalen Menschen und unbeschränkt entfalteten Individualität, Internet: http://www.kritische-informatik.de/pksubj.htm.

O'Reilly & Associates Inc. (1999), Open Source, kurz und gut, Köln: O'Reilly.

Raymond, E. S. (1997), The Cathedral and the Bazaar, http://www.tuxedo.org/~esr/writings/homesteading/cathedral-bazaar/, deutsche Übersetzung: Die Kathedrale und der Basar, http://www.linux-magazin.de/ausgabe/1997/08/Basar/basar.html.

Raymond, E. S. (1998), Homesteading the Nooshpere, http://www.tuxedo.org/~esr/writings/homesteading/homesteading/, deutsche Übersetzung: http://www.phone-soft.com/raymondhomesteading/htn_g.0.html.

Raymond, E. S. (1999), The Magic Cauldron, http://www.tuxedo.org/~esr/writings/homesteading/magic-cauldron/, deutsche Übersetzung: Der verzauberte Kessel, http://www.phone-soft.com/raymondcauldron/cauldron.g.01.html.

Stallman, R.M. (1984), The GNU Manifesto, http://www.gnu.org/gnu/manifesto.html, deutsche Übersetzung: Das GNU-Manifest, http://www.gnu.de/mani-ger.html.

Stallman, R.M. (1994), Why Software Should Not Have Owners, http://www.gnu.org/philosophy/why-free.html.

Stallman, R.M. (1996), Selling Free Software, http://www.gnu.org/philosophy/selling.html.

Stallman, R. (1999), »Software muß frei sein!« Interview des Online-Magazins Telepolis, http://www.heise.de/tp/deutsch/inhalt/te/2860/1.html.

5.3. Anmerkungen

[1] Der »Proprietär« ist der »Eigentümer«, und »proprietär« als Adjektiv heißt so viel wie »einem Eigentümer gehörend«. Doch der Begriff wurde »rundgeschliffen« in Richtung auf »nicht offengelegte Schnittstellen und Datenformate«. Das sind jedoch nur einige Mittel (neben Knebel-Lizenzen, Patenten etc.), um die Exklusivität des Eigentums zu sichern - das alles nur, um künstlich Knappheit zu erzeugen bei einem Produkt, was reichlich vorhanden, weil leicht kopiertbar ist.

[2] So werden auch die »Perversionen« des Kapitalismus erklärlich: Obwohl in vielen Bereichen genug Güter zur ausreichenden Versorgung der Menschheit da wäre, gibt es Armut. Nur wo Knappheit herrscht, ist Tauschwert realisierbar. Der Regulator ist das Geld - wo keines ist, herrscht Armut.

[3] Solche Standards werden in informellen Dokumenten mit dem Titel Request for Comments (RFC) aufgeschrieben. Ihre hohe Verbindlichkeit resultiert aus ihrem offenen Charakter (etwa im Gegensatz zu einem Patent) und der breiten Konsensbildung.

[4] Als Quelltext oder Sourcecode bezeichnet man den von Menschen les- und änderbaren Programmtext. Das maschinenausführbare Programm wird dagegen Binärcode genannt.

[5] GNU ist ein rekursives Akronym und heißt **G**NU Is **N**ot **U**NIX. Es drückt aus, daß das freie GNU-System funktional den proprietären Unix-Betriebssystemen entspricht, jedoch nicht wie diese proprietär, sondern frei ist.

[6] Um freie Software-Bibliotheken auch in nicht-freier Software benutzen zu können, wurde die GNU Library GPL geschaffen, die diese Vermischung erlaubt (z.B. die GNU C-Library). Mit Version 2.1 wurde sie umbenannt in GNU Lesser GPL, vgl. http://www.gnu.org/copyleft/lesser.html.

[7] Tanenbaum, ein Professor aus Amsterdam, veröffentlichte bereits 1986 sein »Mini-UNIX«, genannt Minix, für Lehrverstaltungszwecke. Es konnte sich nie über den Hörsaal hinaus durchsetzen, da es einer restriktiven Lizenz unterlag und die Entwicklung nur von Tanenbaum selbst betrieben wurde. Dokumentiert z.B. in DiBona, C., Ockman, S., Stone, M. (1999) im Anhang A oder im Internet unter http://www.lh.umu.se/~bjorn/mhonarc-files/obsolete/.

[8] Linus Torvalds in einem Interview mit der Tokyo Linux Users Group: »Linux unter die GPL zu nehmen, war das beste, was ich je getan habe.« (O'Reilly & Associates Inc. 1999, 35).

[9] Ich verzichte auf eine differenzierte Darstellung der Einzelaspekte einer »korrekten« Wertformanalyse.

[10] Jegliche Produktherstellung umfaßt einen algorithmisch-konstruktiven und einen operativ-materialisierenden Aspekt. Bei Software geht der Anteil des zweiten Aspekts gegen Null. Mehr zum Thema Algorithmus in Meretz (1999a).

[11] Bill Gates hat QDOS für 50.000 Dollar gekauft unter dem Namen MS-DOS vermarktet, wodurch der Aufstieg von Microsoft begann.

[12] Nach dem 'Chaos Report' der Standish-Group (http://standishgroup.com/visitor/chaos.htm) werden nur ein Viertel aller Projekte erfolgreich abgeschlossen. Der Rest scheitert komplett oder wird mit Zeit- und Budgetüberziehungen von 200% zu Ende gebracht.

[13] Es ist schon lustig, wenn »Freiheit« als ehemaliger Kampfbegriff des Kapitalismus gegen den »unfreien« Sozialismus nun zur Bedrohung im eigenen Hause wird. Anscheinend handelte es sich hierbei auch um zwei »verkrachte Geschwister« - mit letalem Ausgang für den einen.

[14] Eine Diskussion der von ESR verwendeten ökonomischen Kategorien sowie seiner Spekulationen über die Motivation der Hacker (»Geschenkökonomie«) kann ich hier nicht vornehmen. Insbesondere die von ESR dargelegten ökonomischen Kategorien sind haarsträubend. So vertauscht er Gebrauchswert und (Tausch-)Wert sowie Wert und Preis nach Belieben. Das tut der Eloquenz seines Plädoyers für die Re-Integration Freier Software in die kybernetische Wert-Verwertungsmaschine keinen Abbruch. Zum Thema »Geschenkökonomie« vgl. Fischbach 1999.

[15] Vgl. Jamie Zawinski, resignation and postmortem, http://www.jwz.org/gruntle/nomo.html.

[16] Debian Social Contract (Gesellschaftsvertrag): http://www.debian.org/social_contract.

[17] Natürlich wären auch Lockangebote auf Basis der GPL denkbar, doch die Öffentlichkeit würde solche Tricks schnell durchschauen, was dem Image des »Lockers« schaden würde. Da ist die Netscape-Lizenz NPL »ehrlicher«, die besagt, daß man den öffentlichen Code jederzeit wieder privatisieren könne.

[18] Vgl. die Besprechungen in der ZEIT http://www.archiv.zeit.de/daten/pages/199951.p-kurz_.html (PRO) bzw. http://www.archiv.zeit.de/daten/pages//199951.p-kurz-contra_.html (CONTRA) oder bei Telepolis: http://www.heise.de/tp/deutsch/inhalt/co/5659/1.html.

[19] Wer das schlicht »nicht glaubt«, dem empfehle ich direkt den Erfahrungsbericht der Betriebsräte von IBM-Düsseldorf als Lektüre (Glißmann 1999).

Quelle: http://www.kritische-informatik.de/lxwertl.htm
Ich bin an Deiner Meinung interessiert, schreib' mir: stefan.meretz@kritische-informatik.de