FREIE SOFTWARE ALS KOLLABORATIVER TEXT

FLORIAN CRAMER

WAS IST FREIE SOFTWARE?

Why discuss Free Software in the context of net arts and net cultures? Seit ungefähr drei Jahren erfreut sich Freie Software – bzw. „Open Source"– großer Aufmerksamkeit nicht nur in Computerindustrie und Informatik, sondern auch in künstlerischen und politischen Netzkulturren. Die Berliner Konferenz *Wizards of OS* versuchte erstmals 1999, Brücken zwischen Freier Software-Entwicklung einerseits und den Künsten, Geistes- und Sozialwissenschaften andererseits zu schlagen, ein Projekt, das vor kurzem erst auf der *CODE*-Konferenz in Cambrigde und der deutschen *Öko-nux*-Konferenz fortgesetzt wurde. Als gemeinsamer Nenner kristallisierte sich von Anfang an die Politik des „Copyleft" und der freien Verfügbarkeit von Code und Wissen heraus.

Ich möchte heute jedoch einen anderen Aspekt fokussieren, indem ich Freie Software (bzw. „Open Source") als eine Netzkultur lese und ihren Code als einen komplexen, kollaborativen Text. Betrachtet man Freie Software-Entwicklung als eine Schriftpraxis, so erweist sie sich als Avantgarde des Schreibens in digitalen Netzwerken, und nicht nur das: Da Freie Software seit mehreren Jahrzehnten den Kern der technischen Infrastruktur des Internets bildet, hat sie sich faktisch ihr eigenes Netzwerk geschrieben.

Was ist Freie Software? „Freie Software" wird oft mit „Freeware", „Shareware" und anderer proprietären Software verwechselt, die kostenlos vertrieben wird – wie z.B. Microsofts *Internet Explorer*, Apples *QuickTime* oder der *Real Player*. In englischen Sprache liegt dieses Verwechselung noch näher, weil das Wort „free" synonym zu „gratis" ist. Die Free Software Foundation `http://www.fsf.org` definiert Freie Software daher als Software die „frei ist im Sinne der freien Rede, nicht im Sinne von Freibier". Zu den bekanntesten Beispielen Freier Software gehört der Linux-Betriebssystemkern, die GNU-Systemsoftware und der Webserver Apache.

Erst seit 1998 konkurriert der Begriff „Open Source", der von einer Gruppe von Software-Entwicklern um den Essayisten und Programmierer Eric

Date: 14.5.2000.

S. Raymond erfunden wurde, mit „Freier Software". Laut dieser Gruppe ist „Open Source" nur ein anderer Name für dasselbe Konzept, der ihm größere Akzeptanz in der Computerindustrie verschaffen soll.[1] So ist die offizielle *Open Source Definition* [**?**] von Raymond und Co. nur eine Paraphrase der älteren *Free Software Guidelines* [**?**] des Debian-Projekts, einer nichtkommerziellen GNU/Linux-Distribution, die ähnlich wie Linux selbst von unbezahlten Freiwilligen entwickelt wird.[2]

Beide Regelwerke lassen sich wie folgt zusammenfassen:

(1) Freie Software ist frei kopierbar.
(2) Nicht nur der ausführbare Binärcode, sondern auch der Programm-Quellcode ist frei verfügbar.
(3) Der Quellcode darf von jedermann modifiziert oder in andere Programme übernommen werden.
(4) Der Einsatz Freier Software unterliegt keinen Restriktionen. Selbst wenn Freier Software für kommerzielle Zwecke eingesetzt wird, entstehen keine Lizenzkosten.
(5) Der Vertrieb Freier Software unterliegt keinen Restriktionen. Freie Software darf deshalb kommerziell vertrieben werden, ohne daß die Entwickler am Gewinn beteiligt werden.

Da dieselben Kriterien auch für „Open Source" gelten, unterscheiden sich „Freie Software" und „Open Source" tatsächlich nicht in ihrer technischen Definition. Dennoch hat jeder der beiden Begriffe spezifische Unzulänglichkeiten: Während „Freie Software" leicht mit Freeware und Shareware zu verwechseln ist[3] „Open Source" hingegen wird oft mit „offenen Standards" – wie z.B. dem HTML-Format und dem http-Protokoll – verwechselt, sowie mit Software wie Suns Java, deren Quellcode zwar öffentlich zugänglich ist, aber restriktiven Lizenzen unterliegt. Besonders wichtig ist die Unterscheidung von „Open Source" und „Freier Software" von offenen Standards. Währen offene Standards verbindliche technische Spezifikationen sind, die von Industriegremien wie der *Internet Engineering Taskforce* (IETF) und dem *World Wide Web Consortium* (W3C) formuliert werden,

[1] Zitat aus Eric S. Raymonds *Frequently Asked Questions about Open Source*: „The Open Source Initiative is a marketing program for free software. It's a pitch for free software on solid pragmatic grounds rather than ideological tub-thumping. The winning substance has not changed, the losing attitude and symbolism have." [**?**]

[2] Sowohl die *Debian Free Software Guidelines*, als auch die *Open Source Definition* wurden von dem Software-Entwickler Bruce Perens verfaßt.

[3] d.h. Software, die nur im Binärcode vorliegt, kostenlos heruntergeladen werden darf und entweder (als Freeware) ohne Lizenzgebühren oder (als Shareware) gegen vergleichsweise geringe Lizenzgebühren benutzt werden darf.

schreiben „Open Source”- bzw. „Freie Software”-Entwickler für den eigenen Bedard und das eigene Vergnügem was sie immer sie möchten; zerstreitet sich eine Entwicklergruppe, kann sich das Projekt in neue Projekte aufspalten und die Codebasis unabhängig voneinander weiterentwickeln.[4] Solche Absplitterungen verbieten sich in der Entwicklung offener Standards.[5]

Da gerade die Mißverständnisse von „Open Source” so weit verbreitet sind, bleibe ich bei dem weniger populären, aber etwas klareren Begriff „Freie Software”’.

Die Geschichte Freier Software. It is not accidental that history of Free Software runs parallel to the history of the Internet. The Internet is built on Unix networking technology to a large extent. Academic institutions could get Unix for a “nominal fee” including its source code in the early 1970s, and it remains to be the historical base or model of the common Free Software operating systems BSD and GNU/Linux.

Es ist kein Zufall, daß die Freie Software sich historisch parallel zum Internet entwickelt. Das Internet basiert weitgehend auf der Netzwerktechnologie des Betriebssystems Unix. Universitäten konnten Unix im Quellcode zu liberalen Lizenzbedingungen in den frühen 1970er Jahren beziehen, und Unix ist auch die technische Grundlage bzw. das Vorbild der bekanntesten Freie Software-Betriebssysteme BSD und GNU/Linux.

Die enge Verwandtschaft von Unix und Internet ist auch heute noch offensichtlich: E-Mail z.B. ist nichts anderes als Unix-Systemkommando *mail*. Eine E-Mail-Adressse des Musters *xy@z.com* setzt sich aus dem zusammen, was früher einmal ein Benutzerkonto auf einem Mehrbenutzer-Computer war und der Netzwerkkennung dieses Computers. Diese Netzwerkkennung mit dem Internet-Domainnamen wird durch das freie Unix-Programm *bind* ermittelt. Das Domain-Namensystem (DNS) schließlich ist nichts anderes als eine Client/Server-Erweiterung der Unix-Systemdatei */etc/hosts*. Seitdem das auf freier Software und offenen Standards basierende Internet proprietäre Netze wie IBMs EARN/Bitnet, Compuserve, den deutschen Btx und das französische Minitel marginalisiert oder ganz verdrängt hat, ist die Unix-Netzwerktechnologie Standard aller heutigen Betriebssysteme.

---

[4] Ein bekanntes Beispiel ist das Textprogramm XEmacs `http://www.xemacs.org`, dessen Codebasis von GNU Emacs `http://www.gnu.org/software/emacs/emacs.htm` abgespaltet wurde.

[5] Die soziale Dynamik und institutionelle Kontrolle offener Standards beschreibt Jeanette Hofmann, *Der Erfolg offener Standards und seine Nebenwirkungen* [**?**].

In den 1970er Jahren zogen Mehrbenutzer-Betriebssysteme studentische „Hacker" am MIT und der University of California at Berkeley an. Das Konzept offener, dezentraler Computernetzwerke und freier Netzwerkbetriebssysteme hat seinen Ursprung in den Informatikinstituten dieser Universitäten. Während die MIT-Hacker ein eigenes Betriebssystem ITS schrieben und die Berkeley-Hacker den originalen Unix-Code verbesserten und erweiterten, hatten sich Anfang der 90er Jahre aus diesen Hacks zwei freie, Unix-kompatible Betriebssystemfamilien entwickelt:

(1) die Familie der BSD-Betriebssysteme mit den freien Versionen NetBSD, FreeBSD und OpenBSD. Sie alle basieren of Code, der ursprünglich in Berkeley unter dem Projektleiter Bill Joy entwickelt wurde.
(2) das GNU/Linux-Betriebssystem. Alle großen sogenannten Linux-Distributionen – RedHat Linux, SuSE Linux, Turbo Linux, Debian GNU/Linux, Mandrake Linux, Corel Linux OS and Caldera OpenLinux, um nur einige zu nennen – basieren auf der GNU-Software, die seit 1984 von der Free Software Foundation (FSF) geschrieben wurde, und auf dem Betriebssystemkern Linux, der seit 1991 unter der Projektleitung von Linus Torvalds entwickelt wird.[6] Die Free Software Foundation wurde von dem ehemaligen MIT-Hacker Richard M. Stallman gegründet, der auch heute noch ihr Vorsitzender ist.

Offene Technologie ist ein Schlüsselfaktor für die Akzeptanz von Computern und Netzwerken: Die offene Architektur des IBM PCs hat seit den 1980er Jahren Computer preiswert und populär gemacht, mit der offenen Architektur des Internet wurden Computernetze populär in den frühen 1990er Jahren. Seit ca. einem halben Jahrzehnt hat Freie Software jedermann anspruchsvolle Unix-Server-Anwendungen verfügbar gemacht, der bereit war, die technischen Details zu lernen. Ob Freie Software ähnlich populär als Mainstream-Betriebssystem für PC-Anwendungen wird und

[6]Unter Freie Software-Entwicklern ist es strittig, ob Linux-basierte Betriebssysteme nur „Linux" oder „GNU/Linux" genannt werden sollten. Um überhaupt lauffähig zu sein, benötigt eine „Linux"-Installation den GNU C-Compiler (gcc), um den Quellcode in eine maschinenausführbare Binärsoftware zu übersetzen, die GNU C-Bibliothek (glibc) als Schnittstelle zwischen dem Linux-Kernel und System- und Benutzerprogrammen, und die GNU-Tools für grundlegende Benutzerkommandos. Obwohl es theoretisch möglich wäre, die GNU-Tools und die glibc durch andere Software zu ersetzen, nutzen alle gängigen „Linux"-Distributionen die Kombination von Linux und GNU-Software. Ich bleibe daher beim Namen „GNU/Linux", sofern ich mich nicht nur auf den Betriebssystenkern, sondern das lauffähige Betriebssystem beziehe.

das Ende proprietärer PC-Betriebssysteme einläutet, halte ich für unwahrscheinlicher, ist aber nicht die Frage, die hier beantworten möchte.

FREIE SOFTWARE ALS NETZKULTUR

Freie Software kann nicht auf eine pragmatische Entwicklungsmethode reduziert werden, sondern lebt vom Idealismus ihrer Entwickler, ein Idealismus, der auch politisch und kulturell motiviert ist. Freie Software ist eine der ältesten Netzkulturen überhaupt. Jüngeren, künstlerisch und politisch orientierten Netzkulturen hat sie ungefähr zwanzig Jahre Erfahrung voraus. Das „Copyleft" Freier Software ist lesbar als Reflexion und juristisch-politische Quintessenz dieser Erfahrung. Erfunden wurde es, um die traditionelle akademische Freiheit der Rede und des Zitierens in die Kultur des Digitalen zu retten. Nichtsdestotrotz hat das Copyleft die traditionell-humanistische Informationsfreiheit radikal neuformuliert. Die Idee, daß Code – also Text – nicht nur frei vervielfältigt werden darf. sondern auch beliebig modifiziert („gepatcht"), recycelt und ohne Genehmigung und Bezahlung des ursprünglichen Verfassers kommerziell vertrieben, ist den neuzeitlichen westlichen Wissenschaften und Künsten völlig fremd. In der Kultur des Buchdrucks sind solche Praxen illegal und werden als Plagiat und Diebstahl angesehen.

FREIE SOFTWARE ALS SCHRIFT

The relevance of Free Software for other net cultures is not limited to the tools it has created and the infrastructures it has made possible, simply because those tools themselves are the very object of Free Software culture: they are *text*, results of complex textual processing. Moreover, this text is being produced with tools which themselves are free code.

Die Bedeutung freier Software für Netzkulturen insgesamt beschränkt sich jedoch nicht auf Lizenzmodelle oder Software-Werkzeuge, die sie der Allgemeinheit zur Verfügung stellt. Denn genau um diese Werkzeuge dreht sich die Kultur der Freie Software: Diese Werkzeuge sind *Text*, Produkte einer komplexen Prozessierung von Schrift. Und dieser Text wiederum wird mit Werkzeugen hergestellt, die selbst Text, freier Code sind.

Zwar trifft die Tatsache, daß Code mit Werkzeugen hergestellt wird, die selbst Code sind, auch auf die Entwicklung proprietärer Software zu. Es gibt aber einen wichtigen Unterschied: Der Text der Freien Software ist der Öffentlichkeit nicht entzogen. Er kann nicht vom Management einer Firma

abgeschafft werden und verschwindet nicht, wenn seine Entwicklung eingestellt wurde. Alle Freie Software akkumuliert zu einer öffentlichen Bibliothek von Programmcode, d.h.: von codiertem Wissen. Sie bildet ein Archiv, ein kulturelles Gedächtnis. Neue Freie Software-Programme müssen daher nicht, wie proprietäre Projekte, bei Null anfangen, sondern können auf der Grundlage dessen gebaut werden, was im öffentlichen Archiv schon vorhanden ist. Freie Software ist daher, um einen literaturwissenschaftlichen Terminus zu benutzen, in hohem Maße intertextuell. Die Entwicklung Freier Software ist die älteste und bis heute erfolgreichste Praxis eines kollaborativen Schreibens in Computernetzwerken. Mit ihrem System textueller Produktion und ihrer Politik des Code ist Freie Software eine textuell weitaus komplexere Literatur als jene, die gemeinhin Netzliteratur oder Netzdichtung genannt wird. [7] Freie Software ist daher zugleich

- ein öffentlich zugängliches, stetig wachsendes Korpus, ein Code-Archiv;
- rekursive (d.h. auf sich selbst angewandte) Textprozessierung, da vorhandener Text sowohl als Quelle *als auch* als Konstruktions-Werkzeug für neuen Code verwendet wird.
- Textprozessierung sogar durch das *Medium* Text, da die Entwicklungsinfrastruktur Freier Software hauptsächlich aus Mailinglisten und kommandozeilen-gesteuerten Versionskontrollsystemen besteht.
- eine „Hacker"-Kultur, die für die Freiheit von Codes und Information einsteht und ihre Politik in die juristischen Texte des Copyleft codiert.

Im codierten Copyleft verbindet sich Freie Software als Netzkultur mit Freier Software als Netztext. Beide Aspekte spielen schon bei der Programmierung Freier Software eine Rolle, die üblicherweise von selbstorganisierten Freiwilligen in Projektarbeit über das Internet geleistet wird. Diese Arbeit besteht aus:

(1) dem Schreiben von Programm-Quelltext

Dazu gehört dazu die Sondierung und Evaluation von verfügbarem freien Quellcode ein, der für das neue Projekt weiterverwendet und adaptiert werden kann. Ebenfalls dazu gehört die Auswahl - und Kompilierung – der Programmierwerkzeuge, die selbst Freie Software-Code sind.

[7] Wie Netzliteratur – „hyperfiction" und "new media poetry"- sich zu poetischen Verfahren verhält, die in Programmiererkulturen entstanden sind, ist näher ausgeführt in meinem Aufsatz [**?**].

Für ihre spezifischen, über das Netz verteilten Schreibweisen haben Freie Software-Entwickler die wahrscheinlich raffiniertesten Schreibwerkzeuge entwickelt. Besonders bemerkenswert ist das *Concurrent Versioning System* (CVS) [**?**], mit dem Autoren Teile eines Quelltext – egal ob er in Programmiersprache oder Umgangssprache verfaßt wurde – per Internet kopieren, offline weiterentwickeln und anschließend ihre Modifikationen mit denen anderer Teammitglieder synchronisieren können. CVS-basiertes Schreiben ist somit der technisch bislang radikalste Schritt weg vom üblichen Textverabeitungsmodell der algorithmisch simulierten Schreibmaschine.

(2) dem Schreiben von Dokumentationstext

Code, der anderen Code dokumentiert, findet sowohl innerhalb, als auch außerhalb des Programmquelltexts, je nachdem, ob er als dessen Annotation geschrieben wird oder als externes Referenzmaterial.

Freie externe Dokumentationen sind Gegenstand einer politischen Debatte innerhalb der Freien Software-Kultur. Einige Firmen machen ihr Geschäft damit, Software unter Freie Lizenzen zu stellen, aber für Dokumentationen und Support Geld zu verlangen.[8] Zumindest im Idealfall gibt es eine zweite textuelle Rekursion in Freier Software, die alle modernen Wissenssystem seit Diderot's und d'Alembert's *Encyclopédie* charakterisiert:[9] Der Code lehrt den Leser alle Arbeitsschritte, die zu seiner Herstellung nötig waren, so das alle enthaltenen Informationen auf sich selbst angewendet werden können.

(3) der Kommunikation über Mailinglisten, Bugtracking-Systeme und IRC Chat

Die Entwicklerteams für Freie Software-Projekt konstituieren und verständigen sich fast ausschließlich über das Internet, d.h. in Mailinglisten, Newsgroups und auf IRC-Servern. Interpersonale Kommunikation bildet daher eine dritte Schicht von Text, die die Gestaltung sowohl des Programmcodes, als auch der Dokumentationstexte reguliert. Sie operiert somit als kybernetische Rückkoppelungsschleife des Entwicklungsprozesses.

(4) dem Schreiben von juristischem Text

---

[8] so z.B. der *O'Reilly*-Verlag, *Sendmail Inc.*, *VA Linux*, *Scriptics*, *Helix Code* und *Eazel*. Alle diese Firmen sind an der Entwicklung oder Dokumentation grundlegender Komponenten von GNU/Linux-Systemen beteiligt.

[9] Ich danke Wau Holland für diese Feststellung, die er auf einem Vorbereitungstreffen der ersten *Wizards of OS*-Konferenz äußerte.

> Freie Software ist ausschließlich juristisch definiert, denn sie ist Software unter bestimmten Lizenzen. Die verbreitesten Copyleft-Lizenzen sind die GNU General Public License `http://www.gnu.org/copyleft/gpl.html`, die BSD-Lizenz und die Perl Artistic License. Ob ein Programm-Quelltext Freie Software ist, ist allein durch die Lizenz definiert. Deshalb ist juristischer Code die vierte Textschicht, die den gesamten Fluß von Text reguliert, der in Freie Software-Projekten geschrieben wurde.

Freie Software ist somit ein hochdifferenziertes System rekursiver Texterzeugung für ein öffentlichen Wissenpool, ein kulturelles Gedächtnis. Sie ist Textcode, der aus Textcode mit textcodierten Werkzeugen und mittels textueller Kommunikation über textuell codierte Netzwerke erzeugt wird. Die Textarten, die als Freie Software prozessiert werden, sind äußerst vielfältig: ausführbare Binärdateien[10], Text in Programmiersprachen, Texte in natürlicher Sprache zur Dokumentation, für die Kommunikation und das Projektmanagement, und schließlich juristische Texte, in denen die Fairplay-Regeln dieser rekursiven Textprozessierung aufgestellt sind

OBJECTIONS

Both the Free Software engineering and the net artistic camps are traditionally skeptical about attempts to read Free Software in terms of the net arts. The objections were particularly voiced when the Linux kernel was awarded the Golden Nica in the "net" category of *Ars Electronica* 1999. At the *Wizards of OS* conference in the same year, the net artist Alexej Shulgin argued that Free Software is "functional" while *Net.art* is "non-functional", self-sufficient code.[11]

I do not find this point viable from an analytical perspective, since the division between "functional" and "non-functional" is purely arbitrary and subjective. I/O/D's *Web Stalker* [**?**], an experimental Web browser and well-known *Net.art* work, is arguably more "functional" than the teddy bear desktop emblem *xteddy* which is contained in all major GNU/Linux distributions. Moreover, the dinstiction between "functional" Free Software and "non-functional" *Net.art* falls back into late-romanticist notions of the

[10]Which can be read as "text" if text is linguistically and semiotically defined as a finite number of discrete signs chosen from a finite set of signs. In computing, "text" is rather colloquially understood as code from natural-language alphabets as opposed to binary code. Being a philologist, I refer to the prior concept of "text".

[11]According to [**?**], the label *Net.art* was coined in 1996 by the net artist Vuk Cosic and has been associated with a particular generation of net artists since (involving, among others, Cosic himself, Heath Bunting, Olia Lialina, Alexej Shulgin, jodi and I/O/D).

absolute artwork versus lower craftsmanship. It also neglects that with its multiple self-applications of text, the development and use of Free Software is to a large extent its own purpose. No other operating system is as open and seductive to be used as an end to itself as GNU/Linux.

Just as arbitrary as the distinction between “functional” and “non-functional” software is that between program source code and poetry. To date, all attempts to formally define poetry and poetic language have failed. The decision whether a text is poetry will always be up to the reader. The notion of “program code” versus “poetry” was first put into question by the French poet and mathematician François le Lionnais, who co-founded the Oulipo group with Raymond Queneau. In 1973, le Lionnais released a volume of poetry written in the programming language Algol. The practice has been revived in the 1990s by people who write poems in the Perl scripting language.

SCHLUSS

Liest man Freie Software als Netzkultur und Netzliteratur, so ist sie ein hochdifferenziertes, komplexes System sozialer Interaktionen und auf sich selbst applizierter Texte. Keine andere Netzkultur hat sich ihren digitalen Code so radikal selbst geschrieben wie die Freie Software, und keine andere Netzkultur war sich so früh der Kultur und Politik digitaler Schrift bewußt.

C/O FREIE UNIVERSITÄT BERLIN, SEMINAR FÜR ALLGEMEINE UND VERGLEICHENDE LITERATURWISSENSCHAFT, HÜTTENWEG 9, 14195 BERLIN

E-mail address: cantsin@zedat.fu-berlin.de