Freie Software - Ressourcen schonen durch Teilen

Inhaltsverszeichnis

Zusammenfassung

Dieser Artikel ist im Zusammenhang des Fachgesprächs *„Nachhaltige Software“* entstanden, veranstaltet am 28.11.2014 durch das Öko-Institut e.V. sowie das Umweltbundesamt. Er ist ein Beitrag zu der Forschungsfrage *„Was ist Nachhaltige Software?“* beziehungsweise zu der Frage, welche Kriterien für die Nachhaltigkeit von Software und deren Entwicklung in Betracht gezogen werden können.

Explizit widmet sich dieser Artikel den nachhaltigen Aspekten Freier Software und offener Entwicklungsmodelle. Dazu wird zuerst in das Verständnis von Nachhaltigkeit und in die Nachhaltigkeit digitaler Ressourcen eingeführt. Es folgt eine Erklärung Freier Software und schließlich wird skizziert, welche direkten positiven ökologischen Auswirkungen durch die Entwicklung und Verwendung Freier Software erzielt werden können.

Autor: Erik Albers
Kontakt: *eal@fsfe.org*

Berlin, Dezember 2014

1. Software im Kontext der Nachhaltigkeit

Je mehr unsere alltäglichen Interaktionen durch die Verwendung digital-technischer Geräte geprägt werden, desto mehr rückt die Ökobilanz dieser technischen Geräte in den Fokus der Aufmerksamkeit. Elektroschrott, Recycling, „Green-IT“, Ökostrom, Halbwertszeit von Hardware und verwandte Aspekte gewinnen an Bedeutung. Über die „Nachhaltigkeit“ der verwendeten Software wird hingegen kaum gesprochen. Was könnte man darunter verstehen, was ist nachhaltig und welche Anforderungen müsste Software erfüllen um nachhaltig zu sein?

1.1 Zum Begriff „Nachhaltigkeit“

Der Begriff „Nachhaltigkeit“ hat im letzten Jahrzehnt eine nahezu inflationäre Verwendung gefunden[1], insbesondere in Politik und Wirtschaft. Als eine Folge daraus wurde in unterschiedlichsten Kontexten von Nachhaltigkeit gesprochen, so dass der Begriff weiter an Unschärfe gewonnen hat. Deshalb wird den weiteren Ausführungen dieses Artikels folgende Definition zu Grunde gelegt:

> *„Dauerhafte Entwicklung ist Entwicklung, die die Bedürfnisse der Gegenwart befriedigt, ohne zu riskieren, daß künftige Generationen ihre eigenen Bedürfnisse nicht befriedigen können.“*

Dieses Zitat ist einer der zentralen und meist zitierten Leitsätze des Brundtland-Berichts *„Unsere gemeinsame Zukunft“* (Brundtland 1987, Absatz 49)[2]. Durch die darin enthaltene intertemporale Dimension wird in Kurzform auch von „Generationengerechtigkeit“ gesprochen.
Bei Betrachtung endlicher Ressourcen, die eines Tage erschöpft sein könnten, wäre unter der Generationengerechtigkeit ein Konsum zu verstehen, der die vorhandenen Ressourcen auf eine Art und Weise schont, dass auch zukünftige Generationen noch von der gleichen Ressource zehren können. Bei Software gilt es, diese Aspekte der Ressourcenschonung beim Energieverbrauch der Software zu bedenken, sowie im Rohstoffverbrauch bei der für den Betrieb der Software notwendigen Hardware. Betrachten wir hingegen Software selbst als Ressource, ergeben sich andere Bedingungen der Generationengerechtigkeit. Denn Software ist digital und damit immateriell – und für den Erhalt digitaler Ressourcen gelten andere Kriterien einer nachhaltigen Entwicklung.

1 Der Autor empfiehlt dazu eine satirische Visualisierung der zunehmenden Verwendung des Begriffes “Sustainable” durch den Webcomic “xkcd”: *http://xkcd.com/1007/*

2 Der sogenannte *“Brundtland-Bericht”* ist der Abschlussbericht der von den Vereinten Nationen eingesetzten Brundtland-Kommission. Seine Veröffentlichung hat maßgebend zu einem gemeinsamen globalen Verständnis einer nachhaltigen Entwicklung beigetragen.

1.2 Nachhaltigkeit digitaler Ressourcen

In der Digitalen Gesellschaft werden unser Arbeitsalltag, unsere Kommunikation und ein großer Teil unserer Infrastruktur inzwischen von Software gesteuert oder vereinfacht. Heute in Deutschland Heranwachsende ohne Zugang zu jeglichem Computer aufzuziehen wäre geradezu verantwortungslos, da es einem sozialen und gesellschaftlichen Ausschluss gleichkäme. Je mehr kritische Infrastruktur von Software gestellt wird, desto wichtiger werden auch der Erhalt, das Verständnis und der Zugang zu dieser Software. Software wird zu einem Fundament, dessen eigene Existenz zum Erhalt unser digital gewachsenen Infrastruktur notwendig ist; ebenso wie das Wissen um die Bedienung der Software. Es ist an der Zeit, Software als Informations- und Wissensressource zu verstehen.

Betrachten wir Software als eine Ressource, unterscheidet sich diese grundlegend von natürlichen, endlichen Ressourcen. Software – sowie digitale Information jeglicher Art – ist eine immaterielle Ressource und unterliegt damit ähnlichen Bedingungen wie Wissens- und Kulturgüter. Für digitale Ressourcen gilt, dass trotz exzessiver Anwendung, Gebrauch und miteinander Teilen kein Verzehr der Güter stattfindet. Jeder kann von derselben Ressource profitieren ohne dabei die Ressource für andere Konsumenten zu schmälern.
Beispiel: Betrachten wir ein einzelnes Computerprogramm als eine Ressourcen-Einheit der Menge Eins, dann kann diese Einheit prinzipiell unendlich oft verwendet, kopiert und getauscht werden ohne dass sich dadurch die Ursprungsressource jemals verbrauchen würde. Im Gegenteil: Mit jeder Kopie des Computerprogramms steigt die Menge der zur Verfügung stehenden Ressourcen-Einheiten. Betrachtet man eine spezifische Form digitaler Information – zum Beispiel ein einzelnes Computerprogramm – als besonders schützenswert und möchte diese Ressource für zukünftige Generationen erhalten, dann ist eine erfolgversprechende Strategie, möglichst viele Kopien davon (= Einheiten) möglichst weit zu verbreiten.
Zusammenfassend gilt, dass Software dann zu einer nachhaltigen Ressource wird, wenn sie frei kopiert und angepasst werden kann. Diese Möglichkeiten müssen jedoch explizit eingeräumt werden. Denn obwohl Software sich durch die Erstellung von Kopien nicht erschöpfen kann, können digitale Ressourcen durchaus ausschließbare Güter sein: Durch Anwendung von Immaterialgüterrechten, geschlossenem Code und Kopierschutzmaßnahmen werden digitale Ressourcen künstlich verknappt und damit vom Teilen, dem freien Gebrauch und der freien Weiterentwicklung ausgeschlossen. Die künstliche Verknappung des Angebots soll Markt erschaffend wirken; allerdings liegt gerade in der künstlichen Verknappung digitaler Ressourcen die Gefahr des unwiderruflichen Ressourcen- und damit Wissensverlustes. Denn mit der Nicht-Veröffentlichung des dem Programm zu

Grunde liegenden Codes wird das Wissen um die Herstellung und Funktionsweise der Software privatisiert. Das sorgt für einen unmündigen Kunden, der vom Support und dem Gutdünken des Herstellers abhängig ist. In unserem Zusammenhang ist es darüber hinaus vor allem eine Gefahr für den Erhalt der Software als zukünftige Ressource.
Durch das Verschließen des Codes liegt das Wissen um die Funktion sowie das Recht der Kopie und der Verbreitung dieser Information in den Händen einer geschlossenen Gruppe, beispielsweise eines Unternehmens. Verschwindet dieses Unternehmen eines Tages vom Markt, dann ist die Gefahr groß, dass damit auch für immer das Wissen um die Software verloren geht. Gleiches gilt für verschlossene Dateiformate. Daten, die in proprietären Formaten gespeichert werden, können nicht als zukunftssicher gelten, denn es sind nur ganz spezielle, verschlossene Computerprogramme fähig, diese Daten auszulesen. Mehr noch: Proprietäre Dateiformate erschaffen und gestalten Insellösungen und Inkompatibilität. Dadurch sollen Kunden an den Hersteller gebunden werden. Geht jedoch dem Hersteller das Wissen um seine proprietären Formate verloren, ist es damit zugleich den zukünftigen Generationen verloren gegangen. Das ist eine der wenigen Gefahren, wie sich eine digitale Ressource in der Tat erschöpfen kann.

Dieser Gefahr entgegen wirken Freie Software[3] und Offene Standards[4]. Beide stehen der Öffentlichkeit dank ihrer offenen Lizenzbestimmungen frei zur Verfügung. Durch öffentliche Dokumentation sowie das explizite Recht, die Software zu kopieren und anzupassen, bilden sie das Fundament, um digitale Information und Software als eine nachhaltige Ressource für zukünftige Generationen zu erhalten. So lange es Computer gibt, werden diese in der Lage sein Freie Software zu verstehen und zu verwenden. Das erfüllt nicht nur den Anspruch der Generationengerechtigkeit, sondern auch einen weiteren Leitsatz des Brundtland-Berichts:

> *Im wesentlichen ist dauerhafte Entwicklung ein Wandlungsprozeß, in dem die Nutzung von Ressourcen, das Ziel von Investitionen, die Richtung technologischer Entwicklung und institutioneller Wandel miteinander harmonieren und das derzeitige und künftige Potential vergrößern, menschliche Bedürfnisse und Wünsche zu erfüllen.“*[5]

Verfolgen wir also eine Entwicklung, welche *„die Nutzung von Ressourcen“* auf eine Weise ermöglicht, *„das derzeitige und künftige Potential [zu] vergrößern“*, dann müssen wir auf Freie Software

3 Für eine Einführung in Freie Software, siehe Kapitel 2: *„Freie Software – Definition, Entstehung und heutige Bedeutung“*

4 Unter dem Begriff *„Offene Standards“* werden Dateiformate oder Protokolle verstanden, deren Dokumentation, Verwendung und Implementation frei, offen und für alle zugänglich ist. Für eine genaue Definition siehe die *„Genfer Erklärung zu Standards und der Zukunft des Internets“* (Genf 2008)

5 Brundtland 1987, Absatz 15

und Offene Standards setzen. Nur diese können die Ressource Software nachhaltig sichern und deren Potenzial auch für die Zukunft vergrößern. Es ist an der Zeit für die Digitale Gesellschaft eine *Digitale Nachhaltigkeit*[6] zu fördern.

1.3 Nachhaltigkeit 'durch' aber auch 'von' Software

Die bisherigen Ausführungen legen nahe, dass bei der Bestimmung von „nachhaltiger Software“ mindestens zwei Aspekte von Nachhaltigkeit berücksichtigt werden sollten. Zum einen die ökologischen Auswirkungen durch den Betrieb von Software, eine Nachhaltigkeit *'durch'* Software. Zum andern das Verständnis von Software selbst als eine Ressource, das eine Nachhaltigkeit *'von'* Software fordert.

Bei der Betrachtung von Nachhaltigkeit *durch* Software ist in erster Linie der Energieverbrauch ein maßgebender Faktor. Um Prozesse durchzuführen, benötigt Software Energie. Energieerzeugung bringt jedoch häufig massive Umwelt- und Klimaeinflüsse mit sich. Eine logische Ableitung aus diesem Zusammenhang wäre, dass nachhaltige Software eine Software sei, die möglichst effizient und dadurch Energie schonend läuft. Betrachten wir als Beispiel Software A und B, welche exakt dieselbe Leistung erbringen, wobei A jedoch weniger Energie für das gleiche Ergebnis benötigt. Dann wäre A in unserem Sinne als eine Energie schonende und damit nachhaltige Software zu verstehen.
Der Energieverbrauch zum Betrieb der Software ist jedoch nur ein Aspekt. Einen weitaus größeren ökologischen Fußabdruck hinterlässt die zum Betrieb von Software produzierte Hardware. Ihre Produktion erfordert hohe Mengen an Energie und oft den Einsatz äußerst begrenzter Ressourcen, z.B. sogenannter „seltener Erden“. Software, die zum Betrieb möglichst geringe Anforderungen an Hardware stellt oder gar eine langfristige Verwendung von Hardware garantiert, ist damit im Sinne der Ressourcenschonung definitiv auch als eine nachhaltige Software zu verstehen.

Schließlich müssen Software und digitale Information selbst als Ressource betrachtet werden. Software wird immer mehr zur kritischen Infrastruktur der Digitalen Gesellschaft. Die größte Gefahr für die Nachhaltigkeit *von* Software besteht – analog zu der Ressource Wissen – darin, die eigentlich unbegrenzte Ressource in einer Weise zu verknappen oder zu verschließen, dass zukünftige Generationen nicht mehr daran teilhaben können. Im Gegenzug können sich insbesondere in der Verwen-

6 “Digitale Nachhaltigkeit” ist ein noch junger Begriff ohne allgemeingültige Definition. In der Wissenschaft (vgl. Busch 2008, Grassmuck 2004, Stürmer 2009) hat sich jedoch eine weitgehend einheitliche Verwendung des Begriffes etabliert (vgl. Martens 2013). Demnach wird unter Digitaler Nachhaltigkeit mindestens verstanden: die Verwendung Freier Software und Offener Standards, der offene Zugang zu Daten und die Freie Zirkulation von Daten.

dung von Software als eine gemeinsame Ressource erhebliche Synergieeffekte ergeben, da effiziente Lösungen geteilt und damit potenziert werden[7]. Wie zu sehen sein wird, kann Freie Software zu allen drei Aspekte – Energieeffizienz, Hardwareschonung und Digitale Nachhaltigkeit – einen entscheidenden Beitrag leisten.

2. Freie Software – Definition, Entstehung und heutige Bedeutung

Bis in die 70er Jahre des 20. Jahrhunderts wurde Software weitgehend als ein freies Gut behandelt und inklusive des offenen Quellcodes verbreitet. Computer sind zu dieser Zeit meist im universitären Umfeld zu finden. Es ist üblich, die zugehörige Software auf ihre Eigenschaften und Auswirkungen hin zu untersuchen, diese zu teilen und zu verbessern. Transparenz, sowie die Wiederverwendung oder Wiederholung von Ergebnissen und Erkenntnissen bilden die Grundlagen der elektronischen Informationswissenschaft.
Mit dem Aufkommen des Heimcomputer in den 70er und 80er Jahren beginnen Hardwarehersteller jedoch damit, ihre für den Betrieb der Hardware mitgelieferte Software nur noch in verschlossenen Dateiformaten und damit ohne den zu Grunde liegenden Quellcode auszuliefern. Das Wissen um den Betrieb der Maschine soll verschlossen bleiben. 1974 wird das amerikanische C*opyright* auch auf Computerprogramme ausgeweitet. Es kommt zur Verbreitung des *Personal Computer*s der Firma *International Business Machines Corporation (IBM)* und dessen mitgeliefertem Betriebssystem *DOS* der Firma *Microsoft.* Damit wird im außeruniversitären Umfeld der Grundstein zur Akzeptanz des Vertriebs *proprietärer Software* gelegt. Als proprietäre Software wird Software bezeichnet, deren Quellcode nicht ersichtlich ist und für die zudem nur eingeschränkte Nutzungsrechte gelten.

Um dieser Entwicklung gegenzusteuern, initiiert Richard Stallman – Mitarbeiter im Labor „Künstliche Intelligenz“ des Massachusetts Institute of Technology – 1983 die Entwicklung des GNP--Projektes[8]. Ziel des GNU-Projektes ist, ein komplettes Betriebssystem zu schreiben, dass vollständig aus *Freier Software* besteht. Damals bis heute wird unter Freier Software jegliche Software verstanden, die allen Nutzenden vier grundsätzliche Rechte (sogenannte „Freiheiten“) einräumt[9]. Diese

7 Zur Erläuterung dieses Arguments siehe Abschnitt 3.3 *„Gemeinsame Lösungen für eine gemeinsame Ressource“*
8 In Anlehnung an das damals weit verbreitete Betriebssystem *Unix* ergibt sich das Akronym *„GNU“*, das bedeutet *„Gnu's Not Unix“*.
9 Die Anwendung und Garantie dieser vier Freiheiten für alle Nutzenden hat bedeutende Auswirkungen auf die Software selbst, deren Entstehungsprozess sowie deren Verwendung und Bedeutung. Diese Aspekte können hier allerdings nicht diskutiert werden. Für mehr Hintergrund siehe *https://fsfe.org/freesoftware*

sind das Recht die Software

1) **zu verwenden** (zu jedem Zweck, ohne Einschränkung)
2) **zu verstehen** (den Quellcode einzusehen um das Programm untersuchen zu können)
3) **zu verbreiten** (das Programm beliebig oft zu kopieren und zu teilen)
4) **zu verbessern** (das Programm zu verändern und die neue Version selbst zu veröffentlichen)

Diese Nutzungsrechte werden gegeben und garantiert durch Lizenzen. Die Lizenz der Software beschreibt, was Nutzende unter welchen Umständen mit der Software machen dürfen. Im Gegensatz zu proprietärer Software räumen die Lizenzen Freier Software explizit das Recht ein die Software zu verstehen und zu teilen. Lizenzen dieser Art gibt es viele. Die *Free Software Foundation (FSF)* führt dazu eine Liste von anerkannten Freie Software Lizenzen[10]. Die am häufigsten verwendeten Lizenzen[11] sind in Reihenfolge ihrer Häufigkeit *GPLv2*, *MIT* , *Apache*, *GPLv3* und *BSD*. Grob lassen sich diese Lizenzen in zwei „Familien" einteilen, die sich durch unterschiedliche Formen der „Weitervererbung" unterscheiden: Manche Lizenzen fordern, dass im Falle einer Veränderung und Wiederveröffentlichung des Quellcodes der abgewandelte Code unter derselben Lizenz veröffentlicht werden muss. Damit wird erreicht, dass eine einmal als Freie Software veröffentlichte Software sowie alle Ableitungen und Weiterentwicklungen frei bleiben. Dieses weitervererbende Prinzip wird als „*Copyleft"* bezeichnet.
Andere Lizenzen hingegen fordern nicht ihre eigene Weitervererbung. Zur Abgrenzung von *Copyleft*-Lizenzen werden diese als *„permissive Lizenzen"* bezeichnet. Wer den Quellcode einer Freien Software verändert, die unter *permissiver Lizenz* veröffentlicht wurde, kann diese Abwandlung entweder unter der gleichen Lizenz veröffentlichen, unter einer *Copyleft*-Lizenz oder gar unter einer proprietären Lizenz. Beide Lizenz-Familien finden in der Praxis etwa gleich häufig Verwendung[12].

1984 kündigt Richard Stallman seinen Job, um sich in Vollzeit der Entwicklung von GNU zu widmen. 1985 gründet er die *Free Software Foundation* und entwickelt 1989 mit der Veröffentlichung der ersten *GNU General Public License (GPL)*[13] das Prinzip des *Copyleft.* Stallmans ursprüngliche Ankündigung der Entwicklung des GNU-Projektes gilt heute als Geburtsstunde Freier Software und der Freien Software Bewegung. Anfang 2000 entwickelt Lawrence Lessig aus den Prinzipien Freier Software und deren Lizenzen die *Creative Commons* Lizenzen, die eine Adoption der Ideen Freier

10 Siehe *https://www.gnu.org/licenses/license-list.html*
11 Siehe Black Duck 2014
12 Siehe Fussnote 11
13 Inzwischen gibt es eine ganze „GPL-Lizenzfamilie". Dazu gehören Versionsnachfolger (GPLv2, GPLv3) sowie Abwandlungen (LGPL, AGPL) und deren Nachfolger.

Software auf die Verbreitung freier Wissens- und Kulturgüter darstellt. Zur selben Zeit und in den Folgejahren berufen sich weltweit immer mehr Bewegungen auf die Prinzipien Freier Software und Freien Wissens, darunter *Open Access*, *Open Data*, *Open Source*, *Open Knowledge* und *Open Educational Resources*.

Freie Software finden wir heutzutage in unzähligen digitalen Geräten, die uns umgeben. Darunter eingebettete Systeme (zum Beispiel Internet-Router), digitalisierte Geräte (zum Beispiel Fernseher, Kühlschrank, Mikrowelle), Taschencomputer (zum Beispiel sogenannte „Smartphones“[14]), Supercomputer[15], der größte Teil der Internet-Hardware (zum Beispiel Server[16]) und mit GNU/Linux[17] auch der Laptop und Desktop. Unter Endanwendern bekannte Software ist zum Beispiel Linux, Firefox, Wikipedia, Android, Open/Libre Office, Apache, Wordpress und viele mehr.

3. Freie Software und ökologische Nachhaltigkeit

Die Veröffentlichung des Quellcodes, dessen Teil- und Wiederverwendbarkeit sind maßgebende Eigenschaften Freier Software. Freie Software kann durch diese Charakteristika als gemeinsame Ressource verwendet werden, wie beispielsweise das Wissen in der Bibliothek. Offenheit und Verfügbarkeit des Quellcodes bergen zudem zahlreiche positive Auswirkungen auf unsere sozialen und wirtschaftlichen Organisationsformen. Diese können hier jedoch nicht weiter ausgeführt werden; dieses Kapitel widmet sich ausschließlich den möglichen ökologischen Gewinnen Freier Software.

3.1 Modularität, Freie Software und Effizienz

Freie Software und Betriebssysteme bestehen üblicherweise aus einer Kombination einzelner Module, die mit Hilfe offener Schnittstellen miteinander kommunizieren und so zu einer Gesamtkomposition zusammengestellt werden. Diese Modularität kann dazu dienen, das verwendete Betriebssystem so schlank und effizient wie möglich den Betriebsanforderungen anzupassen. Viele Computer und Computerarbeitsplätze werden schließlich nur dazu verwendet, Texte zu schreiben und zu verarbeiten, sowie diese über das Internet auszutauschen (im Folgenden „Textverarbeitungsplatz“). Wird für einen solchen Textverarbeitungsplatz ein proprietäres System „von der Stange“ erworben, dann ist dieses System üblicherweise dazu ausgelegt viele weitere Aufgaben erledigen zu

14 Das Freie Software Betriebssystem Android hat einen Marktanteil auf europäischen Mobilgeräten von 73,9%. (vgl. ZDNet 2014a)
15 97% der Top 500 schnellsten Supercomputer laufen unter Linux (vgl. ZDNet 2014b)
16 58% aller Webserver laufen mit der Freien Software Apache (vgl. W3Techs 2014)
17 Als *„GNU/Linux“* werden Betriebssysteme bezeichnet, die eine Kombination aus GNU Software und dem Linux-Kernel sind. Diese werden klassischerweise auf dem Desktop und Laptop eingesetzt.

können und erfordert und verwendet dazu zahlreiche Hardwarekomponenten sowie erhöhte Mindestanforderungen an die Hardware. Meist bieten die Vertreiber keine Anpassungen oder bedarfsorientierte Abstufungen ihrer Systeme an. Weil es sich um proprietäre Software handelt, haben Nutzende zudem keine Möglichkeit, das System selbst zu verschlanken oder gewisse Funktionen auszulassen. Bei Verwendung derartiger Systeme entsteht eine Herstellerabhängigkeit, in welcher der Software-Hersteller allen Nutzenden die zur Anwendung benötigte Hardware vorschreiben kann. Bei der Verwendung eines Freien Software Betriebssystems hingegen besteht prinzipiell die Möglichkeit, jedes Modul und Programm manuell zu konfigurieren, zu entfernen, auszutauschen oder auch einzubauen. Da Freie Software geteilt werden darf, können eigens konfigurierte Systeme verbreitet werden und alle Nutzenden von dieser Konfiguration profitieren. Das führt zu Interessensgruppen – sogenannte *communitys* - die spezielle GNU/Linux-Software Konfigurationen – sogenannte *Distributionen* – pflegen und veröffentlichen. Darunter gibt es auch Distributionen, die sich als besonders Hardware- und Ressourcenschonend hervortun[18]. Für den oben angeführten Textverarbeitungsplatz sind solche Distributionen im Sinne der Effizienz bestens geeignet. Ganz generell ermöglicht die Modularität Freier Software ein für die jeweiligen Aufgabenbereiche zugeschnittenes System zu erstellen, ohne unnötigen Ballast. Oder, wie es Antoine de Saint-Exupéry einst ausdrückte: *"Perfektion ist nicht dann erreicht, wenn es nichts mehr hinzuzufügen gibt, sondern wenn man nichts mehr weglassen kann."*[19]

3.2 Hardware als Ressource verstehen

Die Produktion von Hardware erfordert jede Menge Ressourcen, sowohl seltene Rohstoffe als auch Energie. Jede Möglichkeit, unseren Hardwareverbrauch zu senken, kann als ein Beitrag zur ökologischen Nachhaltigkeit betrachtet werden. Darunter wäre eine bedarfsorientierte Neuanschaffung von Hardware zu verstehen sowie eine möglichst lange Verwendung alter Hardware. Wie soeben ausgeführt, bietet Freie Software, insbesondere GNU/Linux-Systeme, dafür speziell zugeschnittene Distributionen, die möglichst wenige Anforderungen an die Hardware stellen. Hinzu kommt, dass der offene Quellcode eigene Anpassungen sowie die Übertragbarkeit der Software auf verschiedene Endgeräte ermöglicht. Das heißt, für unser Beispiel des Textverarbeitungsplatzes benötige ich mit der Verwendung Freier Software nicht länger einen Laptop oder Desktop-Computer. Durch die Ver-

18 *thinkwiki.de* listet unter „*Ressourcenschonende Linux Distributionen*“ (thinkwiki 2014) beispielsweise *Debian* (eine Distribution) für den Einsatz mit einer *i486*-CPU-Architektur. Das ist Hardware, die in Desktops zwischen Anfang und Mitte der 90er Jahre eingebaut wurde. In Kombination mit einer ressourcenschonenden Desktopumgebung, beispielsweise *LXDE* oder *Xfce* (siehe thinkwiki 2014), kann damit auf bereits 20 Jahre alter Hardware eine aktuelles Betriebssystem mit grafischer Oberfläche effizient betrieben werden.

19 Saint-Exupéry 1939: S. 60

wendung eines schlanken GNU/Linux-Systems mit grafischer Oberfläche und einer Libre OfficS--Suite[20] ist ein komfortabler Textverarbeitungsplatz bereits mit viel geringerem Hardwareaufwand möglich, beispielsweise mit dem Mini-Computer *RaspberryPi*[21]. Anstatt sich also vom Hersteller diktieren zu lassen, welche Hardwareanforderungen dessen multifunktionales Betriebssystem benötigt, kann durch die Verwendung Freier Software ein möglichst schlankes System verwendet werden und in Folge dessen eine bedarfsorientierte Neuanschaffung von Hardware erfolgen.

Die Unabhängigkeit der Software und ihrer Anwendung wirkt sich zudem positiv auf den Lebenszyklus und die Langlebigkeit von Hardware aus. Mit Freier Software kann ökologisch problematischen Geschäftsmodellen entgegengewirkt werden, welche vorsehen die Halbwertszeit von Hardware gering zu halten oder den Wert alter Hardware zu senken, indem neue Softwareentwicklungen nicht länger den Einsatz alter Hardware unterstützen. Denn selbst viele Jahre alte Rechner sind meist vollkommen ausreichend um mit Hilfe schlanker Distributionen einen stabilen, schnellen und effizienten Textverarbeitungsplatz zu ermöglichen. Die Unabhängigkeit und Anpassungsfähigkeit Freier Software bietet dadurch die Freiheit, selbst zu entscheiden, ab wann eine Hardware zu alt für den eigenen Einsatz geworden ist.
Um den Vertrieb neuer Hardware zu fördern, wird teilweise versucht zu argumentieren, dass neue Hardware doch viel Energie sparsamer sei als alte Hardware. Das ist allerdings Augenwischerei, denn die Herstellung neuer Hardware zehrt in der Energiebilanz jeden Effizienzgewinn auf. Ökologisch schneidet jede Wieder- oder Weiterverwendung von Hardware besser ab als Elektroschrott, selbst mit Recycling. Auch birgt die Verwendung alter Hardware eine soziale Komponente. Wer für den eigenen Betrieb neue Hardware benötigt, kann die ausrangierte Hardware beispielsweise noch sinnvoll als Arbeitsplatz in anderen Abteilungen, in Bildungseinrichtungen oder in der Entwicklungshilfe einsetzen.

3.3 Gemeinsame Lösungen für eine gemeinsame Ressource

Freie Software gibt jeder und jedem das Recht, die Software frei zu verwenden, zu verstehen, zu verbreiten und zu verbessern. Dadurch entsteht ein Entwicklungsmodell, welches es uns ermöglicht, Software als eine gemeinsame Ressource zu verstehen und zu verwenden. Einzelne Lösungen können so für gemeinsame Lösungen sorgen, Effizienzgewinne sich dadurch potenzieren.

20 Libre Office wird hier nur als ein Beispiel für eine komplette Büroanwendungsumgebung auf Basis Freier Software verwendet. Dieselbe Aussage gilt auch für andere Büroanwendungssoftware, die als Freie Software lizenziert ist.

21 *„RaspberryPi“* bezeichnet einen preisgekrönten Einplatinencomputer, der besonders günstig (derzeit in etwa 40 Euro) und zudem energiesparsam ist. Seit 2012 kann dort ein komplettes Libre Office-Systen zum Laufen gebracht werden (siehe TDF 2012)

Ein Beispiel: Der Stadt Berlin gelingt es, eine äußerst effiziente und maßgeschneiderte Software für ihre Verwaltung zu schreiben. Sie beschließt, diese Software per Lizenz als Freie Software zu veröffentlichen. Die Stadt Konstanz bedient sich nun dieser Freien Software und verwendet sie für ihre eigene Verwaltung. Weil die Stadt Konstanz jedoch viel kleiner ist, werden bestimmte Module nicht benötigt, zum Beispiel die Erfassung des Straßenbahnnetzes. Die entsprechenden Module werden entfernt und in Konstanz wird nun eine schlankere Version der Software zum Einsatz gebracht. Nachdem Konstanz ihre selbst angepasste Version wiederveröffentlicht, können andere Kleinstädte davon ebenso profitieren ohne dazu erst eigene Personalressourcen einsetzen zu müssen. Schließlich fällt einem pfiffigen Mitarbeiter in Neubrandenburg beim Betrachten des Codes eine Möglichkeit auf, verschiedene Rechenprozesse zusammenzuführen und die Software damit noch effizienter zu gestalten. Nach erfolgreicher Implementierung und Wiederveröffentlichung durch die Stadt Neubrandenburg können nun auch Konstanz und Berlin die verbesserte Version des Codes in ihren eigenen Anwendungsumgebungen implementieren und somit profitieren am Ende alle Anwender von der gleichen Lösung. Schließlich entsteht so eine gemeinsame Ressource und im Falle von nur einer nachhaltigen Lösung kann sich zugleich die Ökobilanz aller Anwendenden erhöhen.
Dieses Prinzip, das hier als theoretisches Beispiel angeführt wurde, finden wir in der Praxis in vielen Freie Software-Projekten[22]. Als herausragendes Beispiel sei der Linux-Kernel angeführt. Der Linux-Kernel ist eine Schnittstelle, welche die Anweisungen der Software in maschinenlesbare Anweisungen an die Hardware übersetzt. 1991 begann Linus Torvalds als einzelne Person den Linux-Kernel zu entwickeln und als Freie Software unter der GPLv2[23]-Lizenz zu veröffentlichen. Schnell haben sich über das Internet weitere Entwickler eingefunden um gemeinsam den Linux-Kernel zu programmieren. Heute ist der Linux-Kernel wohl der meistverwendete Kernel weltweit. Wir finden dessen Anwendung in nahezu allen Formen digitaler Geräte, vom Router zum Kühlschrank über das Smartphone und den Laptop hin zu Supercomputern[24]. Ermöglicht wird dies durch inzwischen hunderte, tausende Programmierer weltweit, die jeder für sich immer weiter zur gemeinsamen Ressource des Linux-Kernels beitragen und diesen verbessern. Darunter sind Studierende und Freiwillige, aber vor allem auch Angestellte globaler IT-Firmen wie Intel, Red Hat, Samsung, IBM oder Google[25]. Firmen, die auf dem freien Markt in Konkurrenz zueinander stehen, erschaffen hier eine gemeinsame Ressource, die wiederum alle – inklusive der Konkurrenz – ausschöpfen können um Kapital zu generieren. Das ist kein Widerspruch sondern kalkuliertes Geschäft. Durch

22 Genau genommen finden wir das *Prinzip* überall. Allerdings sind in der Praxis manche der Projekte so klein, dass es nur eine Person ist, die tatsächlich Codezeilen beiträgt und die Software somit verbessert.
23 Die 1991 veröffentlichte Folgeversion der ursprünglichen *Gnu General Public License*
24 Siehe dazu auch Ende des Kapitel 2: *„Freie Software – Definition, Entstehung und heutige Bedeutung“*
25 Für die Version 3.18 des Linux Kernel haben allein die oben gelisteten Firmen zusammen 29% aller Code-Änderungen beigetragen (LWN 2014)

die gemeinsame Ressource vermeiden sie die Gefahr, dass Eigenentwicklungen scheitern oder obsolet werden, sie profitieren zudem von den Entwicklungen anderer und müssen nicht jeden Gedanken aufs neue Erfinden oder imitieren.
Ob gewollt oder nicht, alle Beitragenden erzeugen mit der Entwicklung des Linux-Kernels eine digitale Nachhaltigkeit der gemeinsamen Ressource „Software Kernel“. Diese zeichnet sich durch stetige Effizienzsteigerung sowie Anpassungsfähigkeit aus. Die Möglichkeit, diese Entwicklungen auf unzählige Endgeräte zu adaptieren, hat nachhaltige Auswirkungen auf die gesamte Sphäre digital-technischer Hardware. Positive und nachhaltige Aspekte für die Umwelt ergeben sich aus gemeinsamen Möglichkeiten der Energiesparsamkeit sowie dem Erhalt von Hardware beziehungsweise der bedarfsorientierten Neuanschaffung von Hardware.

4. Ausblick

Für die Beantwortung der Forschungsfrage *„Was ist Nachhaltige Software?“* beziehungsweise der Kategorisierung der Nachhaltigkeitskriterien von Software spielt Freie Software und dessen Entwicklungsmodell zweifelsfrei eine wichtige Rolle. Wie gezeigt, fördert Freie Software eine digitale Nachhaltigkeit in der digitalen Gesellschaft und ist die Grundvoraussetzung dafür, Software als gemeinsame Ressource zu verstehen und zu erhalten. Ökologisch können durch Modularität und Anpassungsmöglichkeiten sowohl Energie als auch Hardware als Ressourcen geschont werden. Schließlich sorgen selbst einzelne Lösungen für gemeinsame Effizienzsteigerungen und eine gemeinsame Nachhaltigkeit. In Bezug auf die in Frage stehende Analogie zu dem „Blauen Engel“ empfehle ich deshalb eine Nachhaltigkeitskategorie „Freie Software“.

Quellenverzeichnis:

Black Duck 2014: Black Duck Knowledge Base: “Top 20 Open Source Licenses”, online: *https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses* (abgerufen am 14.12.2014)

Brundlandt 1987: Report of the World Commission on Environment and Development (1987): *“Our Common Future”*. Vereinte Nationen, online: *http://www.bne-portal.de/fileadmin/unesco/de/Downloads/Hintergrundmaterial_international/Brundtlandbericht.File.pdf* (abgerufen am 13.12.2014)

Busch 2008: Busch, Thorsten (2008): *“Open Source und Nachhaltigkeit“* in: Lutterbeck, Bernd / Bärwolff, Matthias / Gehring, Robert (Hrsg.) *„Open Source Jahrbuch 2008“*, Berlin, Lehmanns Media: S. 111 - 122

Genf 2008: Declaration (2008): *„Standards and Future of the Internet.“* Genf, online: *http://www.openforumeurope.org/library/geneva/declaration/manifesto-with-logos-final.pdf (abgerufen am 12.12.2014)*

Grassmuck 2004: Grassmuck, Volker (2004): *„Freie Software – Zwischen Privat- und Gemeineigentum“*, 2. Aufl., Bonn, Bundeszentrale für Politische Bildung

LWN 2014: Corbet, Jonathan (2014): *„Some 3.18 development statistics”*, LWN.net, online: *http://lwn.net/Articles/620827/* (abgerufen am 14.12.2014)

Martens 2013: Martens, Key-Uwe *„Digitale Nachhaltigkeit“* in: Kegelmann, Jürgen / Martens, Key-Uwe (Hrsg.): „Kommunale Nachhaltigkeit“, Nomos Verlag: S. 304 – 314

Saint-Exupéry 1939: Saint-Exupéry, Antoine de (1939) *Terre des Hommes, III: L'Avion*, S. 60, zitiert nach: *https://de.wikiquote.org/wiki/Perfektion* (abgerufen am 13.12.2014)

Stürmer 2009: Stürmer, Matthias (2009) *“Digitale Nachhaltigkeit – ein Konzept mit Zukunft”* Netzwoche #20, online: *http://www.digitale-nachhaltigkeit.ch/2009/11/ein-konzept-mit-zukunft/ (abgerufen am 1.12.2014)*

thinkwiki 2014: Thinkwiki (2014) „Linux Distributionen für ältere Thinkpads“, online: *http://thinkwiki.de/Linux_Distributionen_f%C3%BCr_%C3%A4ltere_Thinkpads (abgerufen am 15.12.2014)*

TDF 2012: The Document Foundation (2012) *“LibreOffice runs on the RaspberryPi“*, online: *http://blog.documentfoundation.org/2012/12/17/libreoffice-runs-on-the-raspberry-pi/* (abgerufen am 12.12.2014)

W3Techs 2014: W³Techs - Web Technologiy Services (2014) *„Usage statistics and market share of Apache for websites“*, online: *http://w3techs.com/technologies/details/ws-apache/all/all* (abgerufen am 14.12.2014)

ZDnet 2014a: Beiersmann, Stefan (2014) *„iOS und Android nehmen Windows Phone in Europa erneut Marktanteile ab“*, ZDnet, online: *http://www.zdnet.de/88209594/ios-und-android-nehmen-windows-phone-europa-erneut-marktanteile-ab/ (abgerufen am 10.12.2014)*

ZDnet 2014b: Vaughan-Nichols, Steven (2014) *„Linux dominates supercomputers as never before“*, ZDnet, online: *http://www.zdnet.com/article/linux-dominates-supercomputers-as-never-before/* (abgerufen am 8.12.2014)