Softwarelizenzen im Vergleich

Freie Software vs. Open Source Software

Manuel Scholz

Dozent: Dr. F. Sarre

Juristisches IT-Projektmanagement
Ludwig-Maximilians-Universität München

Wintersemester 2016/17
15. Januar 2017

Abstract

Freie Software und Open Source Software weisen inhaltlich viele Gemeinsamkeiten vor, und dennoch möchte besonders die Free Software Foundation, dass die beiden Initiativen voneinander getrennt werden. In der Öffentlichkeit hat sich Open Source als Begriff für quelloffene Software durchgesetzt, obwohl einst die „Initiative der Freien Software“ ein Umdenken in der Gesellschaft bewirkt hat. Beide haben sich das Ziel gesetzt, die Welt mit offenen Softwareprojekten zu bereichern. Vor allem für Unternehmen hat der Vertrieb von Open Source Software einige Nachteile, die bei proprietärer Software in dieser Form nicht auftreten. Der Einsatz von Open Source Software lohnt sich für Privatleute und Unternehmen allemal, da teure Lizenzverträge ausbleiben und sie die Software für ihren Zweck frei anpassen können. Allerdings sind dafür die Lizenzbedingungen zu beachten, wie die zwei Beispiele von den Firmen TomTom und Skype am Ende der Ausarbeitung zeigen.

Abkürzungsverzeichnis

BSD Berkeley Software Distribution . . . 3
FLOSS Free Libre Open Source Software . . . 9
FOSS Free Open Source Software . . . 9
FSF Free Software Foundation . . . 3
GCC GNU C-Compiler . . . 3
GPL General Public License . . . 11
GFDL GNU Free Documentation License . . . 12
IIS Internet Information Service . . . 4
LGPL Lesser General Public License . . . 11
MPL Mozilla Public License . . . 5
MIT Massachusetts Institute of Technology . . . 3
OSI Open Source-Initiative . . . 5
OSD Open Source-Definition . . . 7

Inhaltsverzeichnis

Kapitel 1

Einleitung

Obwohl fast jeder (meist unbewusst) mit freier Software arbeitet, um beispielsweise im Internet zu surfen, Bilder zu bearbeiten oder um eine wissenschaftliche Arbeit zu schreiben, wird Software heutzutage häufig mit einer geschlossenen Lizenz ausgeliefert, welche es nicht vorsieht, den Quellcode zur Software mitzuliefern. Diese Art von Veröffentlichung wird unter dem Begriff proprietäre Software zusammengefasst.
Der Gegensatz dazu wird unter dem Begriff Open Source bzw. Free Software verstanden, auf dessen Geschichte, Merkmale und Ausprägungen in dieser Ausarbeitung eingegangen wird. Dabei handelt es sich um kostenlose und quelloffene Software, die vor allem durch das Betriebssystem Linux an Bekanntheit gewonnen hat.
Um quelloffene Software zu unterstützen und zur Einsparung öffentlicher Gelder, die für kommerzielle Software ausgegeben werden müssen, entschied sich die Landeshauptstadt München 2003 für eine Umstellung der Arbeitsplatzrechner von Microsoft Windows auf das Betriebssystem Linux. Die Stadt verspricht sich von „LiMux“ eine größere Herstellerunabhängigkeit und eine Verkürzung der Bearbeitungszeit. [Lan11] Nach einer zehnjährigen Umstellungszeit kamen mit der Zeit vermehrt kritische Stimmen gegenüber der Linux Distribution auf. Ein Gutachter[1] empfahl sogar den Umstieg von LiMux auf das vormals verwendete Betriebssystem Windows. [Kre16]
Dieses Beispiel zeigt, dass freie Software kein Allheilmittel im Kampf gegen proprietäre Software ist. Deswegen behandelt die Ausarbeitung auch die Vor- und Nachteile freier Software und mögliche Konflikte, die bei der Verwendung von freiem Quellcode in der eigenen Software entstehen können.

[1] Das komplette Gutachten ist online einsehbar unter https://www.ris-muenchen.de/RII/RII/DOK/SITZUNGSVORLAGE/4262725.pdf

Kapitel 2

Zeitlicher Ablauf

Um die zeitlichen Zusammenhänge wichtiger Ereignisse mit der Entstehungsgeschichte der Lizenzen verstehen zu können, beginnt der folgende Absatz zu jener Zeit, als Software als eigenständiges Produkt noch nicht weit verbreitet war (vgl. Abbildung 2.1 auf Seite 4).
Bis Ende der sechziger Jahre war Software kostenlos und wurde als Teil des Rechners zusammen mit der Hardware ausgeliefert. Zur damaligen Zeit war es so gut wie nicht vorstellbar, Software als eigenständiges Produkt anzusehen und damit Geld verdienen zu können. Da die Computerindustrie nur aus Hardwareunternehmen (damals dominierend waren IBM, HP, DEC und Data General [Gra04]) bestand, wurde ihre Hardware nur mit Software vermarktet. Deswegen wurde der Quellcode der Software ohne Auflagen weitergegeben. Davon profitierten auch andere Hardwareunternehmen, weil ein Algorithmus nur einmal implementiert werden musste und daraufhin von anderen Entwicklern wiederverwendet und verbessert werden konnte. Im Laufe der Zeit hatten die Unternehmen – auch mit Hilfe der Nutzergemeinde – die Software auf ihre Hardware so speziell zugeschnitten, dass sie die Kunden an ihre Produkte gebunden hatten. Für den Kunden wäre es aufgrund von erforderlichen Schulungsmaßnahmen nur mit viel Aufwand und Kosten möglich gewesen, den Hersteller zu wechseln. Somit konnten die Unternehmen, die Preise für die Hardware für bestehende Kunden stetig steigen lassen. Durch Dumpingpreise war es möglich, weitere Neukunden zu gewinnen. Erst als die amerikanische Kartellbehörde ab 1969 die Marktsituation prüfte, änderte sich etwas an dieser Vertriebsform. Ausschlaggebend war hierbei IBM, das daraufhin freiwillig die Verkäufe der Software von der Hardware abkoppelte. Infolgedessen begannen vermehrt Entwickler ihre Software unter einer kommerziellen Lizenz zu vertreiben. [Teu07, Gra04]
Ein großer Bestandteil des heutigen Erfolges von quelloffener Software ist auf die Entwicklung des Betriebssystems Unix zurückzuführen, welches als Grundlage der heute bekannten Unix-basierenden Betriebssysteme gilt. Die Entwicklung begann in den 60er Jahren in den Bell Laboratorien der amerikanischen Telefongesellschaft AT&T. Als 1969 die erste Version fertiggestellt wurde, wollte AT&T das Betriebssystem vermarkten. Allerdings besaß AT&T eine Monopolstellung am amerikanischen Telefonmarkt, womit durch ein Urteil untersagt war, auch in anderen Wirtschaftszweigen (und damit auch im Softwarebereich) tätig zu werden.

Durch die eigene interne Nutzung von Unix stand es auch im eigenen Interesse von AT&T das Betriebssystem weiter zu verbessern. Demzufolge wurde Unix kostenlos an Universitäten und staatlichen Organisationen weitergegeben. Binnen kurzer Zeit fanden sich hauptsächlich unter den Studenten und Informatikern viele Anhänger, die von dem neuartigen System begeistert waren und daran Verbesserungen vornahmen. Die kalifornische Universität in Berkeley erkannte das Potential des neuen und unkontrollierten Entwicklungsprozesses zuerst und koordinierte den Austausch von Patches, Fixes und Verbesserungen. Daraus entstand 1977 die erste Berkeley Software Distribution (BSD). [Teu07, Sch03]
Im Zuge dessen begannen auch kommerzielle Unternehmen (z. B. Microsoft, Sun Microsystems) Distributionen von Unix zu erstellen und diese zu vermarkten. Nach Grassmuck führte diese Entwicklung dazu, dass Anfang der 80er Jahre nahezu alle Software unter einer proprietären Lizenz[1] stand [Gra04, S. 221 f.].

2.1 Freie-Software-Bewegung

Die zunehmende Kommerzialisierung, fehlende Kooperation zwischen Entwicklern und seine negativen Erfahrungen mit geschlossener Software in seinem Beruf am „Massachusetts Institute of Technology (MIT) für künstliche Intelligenz und Programmierung“ waren unter anderem Gründe für Richard Stallman 1984 das GNU-Projekt ins Leben zu rufen. Dessen Aufgabe war es, ein freies Betriebssystem zu entwickeln, welches als Ersatz für Unix gelten sollte. Durch das rekursive Akronym „GNU’s Not Unix“ wird auf der einen Seite die Ähnlichkeit und auf der anderen Seite die Unabhängigkeit zum nunmehr kommerziellen Betriebssystem Unix aufgezeigt. [J.T01] Da Stallman nicht auf den proprietären Quellcode von Unix zugreifen konnte, musste er diesen komplett von Grund auf neu schreiben. Seine Idee war, zunächst unabhängige Komponenten zu implementieren, um diese am Ende zu einem Betriebssystem zu vereinen. Jedoch arbeitete er zu dieser Zeit auch beim MIT, dessen Anstellung er für das GNU-Projekt aufgeben musste, damit seine Entwicklungen Freie Software bleiben und nicht in den Besitz des MIT wechselten, das daraus wieder proprietäre Software machen würde. Als erste Komponenten implementierte er zunächst den Texteditor GNU Emacs und den GNU C-Compiler (GCC). Ohne festes Gehalt sah sich Stallman gezwungen, Emacs zusammen mit Handbüchern zu vermarkten. Eine freie Emacs Version war aber noch immer vorhanden.
Um das GNU-Projekt auf Dauer finanziell als auch juristisch zu unterstützen, gründete Stallman ein Jahr später die übergeordnete gemeinnützige Organisation Free Software Foundation (FSF). Im Jahr 1990 wurde das Toolkit für das Betriebssystem fertiggestellt. Für das fertige Betriebssystem fehlte nur noch der Kernel, dessen Entwicklung unter dem Namen „GNU Hurd“ eher schleppend voranging. Gründe dafür waren laut Stallman eine sehr aufwändige Fehlersuche, da der Kernel in viele kleinere Programme geteilt wurde und so die asynchrone Kommunikation

[1] Proprietäre Software ist das Gegenstück zu Freier bzw. Open Source Software. Das Kopieren, Studieren, Verändern und Weitergeben des Quellcodes ist untersagt. Deswegen spricht man auch häufig von „Closed Source“.

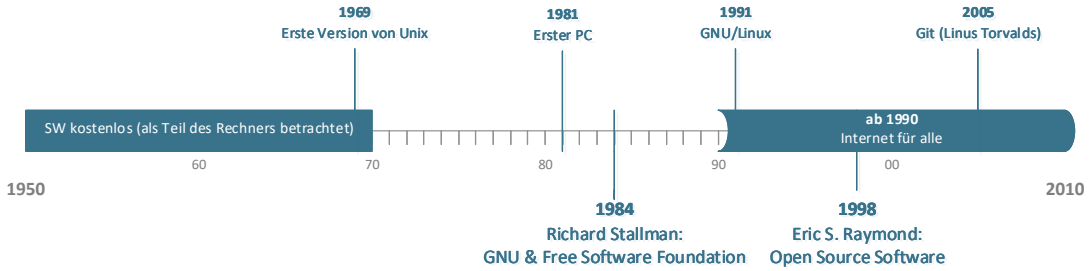


Abbildung 2.1 Zeitliche Einordnung wichtiger Ereignisse für Freie- & Open Source Software

schwer nachzuvollziehen war. Das Resultat war, dass es mehrere Jahre gedauert hätte, den Kernel ins Betriebssystem zu integrieren. Hingegen konnte der unabhängig vom GNU-Projekt zur gleichen Zeit vom finnischen Studenten Linus Torvalds entwickelte Linux-Kernel bereits 1991 fertiggestellt werden. Der Grund für die Entwicklung des Kernels von Torvalds war, dass er während seines Studiums an der University of Helsinki in der Lage sein wollte, ein ähnliches System auf seinem eigenem Computer zu verwenden, wie jenes, das er von den Universitätscomputern (damals SunOS) gewohnt war. Da er kein solches System finden konnte, beschloss er seinen eigenen Kernel zu entwickeln, wobei viele Ideen von SunOS inspiriert waren. Der Kernel war allerdings ohne die im GNU-Projekt entwickelten Programme nichts wert. Demnach bedienten sich erste Benutzer des Linux-Kernels an den freien Paketen des GNU-Projekts, wodurch letztendlich das Linux-System entstand.
Für das GNU-Projekt kam nun das Problem auf, dass Millionen von Personen Pakete von GNU verwendeten, ohne dass sie es wussten. Infolgedessen ergab sich ein Namensstreit, da für Stallman und seine Entwickler die GNU-Pakete eine große Rolle für die Fertigstellung von Linux spielten. Sie pochten auf den gemeinsamen Namen GNU/Linux, um beide Projekte im Namen zu würdigen. Viele Linux-Distributionen sind später diesem Wunsch nachgegangen und bezeichneten ihre Distribution als GNU/Linux (z. B. Debian), wobei andere Distributionen weiterhin nur den Namen Linux verwenden (z. B. Ubuntu).
Durch den Apache-Webserver aus dem Jahr 1993 verbreitete sich Linux sehr schnell weiter. Zum einen war der Einsatz kostengünstiger als die damalige Alternative „Internet Information Service (IIS)“ von Microsoft, zum anderen wurde zur gleichen Zeit das Internet zu einem Massenprodukt, wodurch sich viele Unternehmen für Linux und Apache entschieden haben. [J.T01]

2.2 Die Kathedrale und der Basar

Im 1997 veröffentlichten Aufsatz „The Cathedral and the Bazaar“ [Ray99] bediente sich Eric Raymond mit der Kathedrale und dem Basar an zwei Metaphern, um zwei Entwicklungsmethoden zu paraphrasieren. Dabei soll der Bau einer Kathedrale, der zentral koordiniert und geplant wird, das zu dieser Zeit konventionelle Modell von Microsoft umschreiben. Der Quellcode ist

nur wenigen Personen zugänglich und wird nicht veröffentlicht. Raymond charakterisiert den Basar, und meint damit das Entwicklungsmodell von GNU/Linux, hingegen als unorganisiertes Durcheinander, das sich dann als eine selbst organisierende Gruppe von Personen herausstellt, die aufgrund ihres regen Austausches und Engagements das Projekt vorantreiben. Da der Quellcode frei zugänglich ist, kann jeder das Projekt durch Änderungen verbessern und von der Gruppendynamik profitieren. [J.T01]

Bei der ersten Version des Dokuments verwendete er noch den geläufigen Ausdruck „Free Software“. Ab 1998 ersetzte er diesen durch den Begriff „Open Source“, unter anderem wegen der Zweideutigkeit im Englischen von „Free“ im Sinne von kostenlos und Freiheit. Obwohl die Intention von Stallman bei der Gründung der FSF zweifelsfrei im Sinne der Freiheit stand, könnten nach Raymond naive Menschen darunter Freeware[2] verstehen. Deswegen war das Ziel von Eric Raymond und Mitgründer Bruce Perens, Free Software durch einen Begriff zu ersetzen, der diese Doppeldeutigkeit beseitigt und einen seriösen Eindruck hinterlässt, um auch eine Verwendung in Unternehmen zuzulassen. Beim Gründungstreffen der Open Source-Initiative (OSI) im Februar 1998 wurde der Begriff offiziell verkündet. [Gra04]

Das Unternehmen Netscape nahm in den Anfängen von Open Source eine große Rolle ein, da es die erste große Firma war, die sich an Open Source beteiligte. Zu dieser Zeit dominierte der Internet Explorer von Microsoft den Markt für Webbrowser. Dieser war zwar kostenlos, jedoch war der Sourcecode nicht frei zugänglich und Microsoft erlaubte keine Mitarbeit anderer Personen. Infolgedessen hatte Netscape mit ihrem Webbrowser „Netscape Communicator“ bedenken, dass Microsoft durch das Monopol mit dem Internet Explorer, die HTTP- und HTML-Standards, von denen das Web abhingen, in ihrem Sinne verändern würden. Mit dieser Macht hätten sie die uneingeschränkte Kontrolle gehabt, Netscape und weitere Konkurrenten vom Server-Markt zu drängen, was gleichzeitig das Ende für Netscape bedeutet hätte, da davon ein großer Teil ihres Umsatzes abhing. Dies hatte zur Folge, dass sich Netscape 1998 dazu entschloss, den Quellcode ihres Webbrowsers unter dem Projektnamen Mozilla und der gleichnamigen Lizenz „Mozilla Public License (MPL)“ zu veröffentlichen[3]. [J.T01]

[2] Software, die kostenlos aber ohne Quellcode veröffentlicht wird.

[3] Einordnung der MPL erfolgt in Kapitel 4.1.3

Kapitel 3

Definition von Free Software und Open Source

Wie aus Kapitel 2 hervorging, sind die Ansichten der „Freien-Software-Bewegung“ und der OSI nahezu gleich. In welchen Punkten sie sich unterscheiden und wann eine Lizenz als Open Source- bzw. Free Software-Lizenz gilt, wird in den im Folgenden behandelten Definitionen verständlich. Zu beachten ist, dass die Definition für sich alleine noch keine Lizenz ist. Jedoch gibt sie den Rahmen und die Anforderungen vor, die eine Lizenz erfüllen muss, damit diese als Open Source-Softwarelizenz angesehen werden kann.

3.1 Free Software-Definition

Die Definition der Freien Software umfasst insgesamt vier Freiheiten (Four Freedoms), die die Lizenz einhalten muss, damit diese als Freie Software gilt. Dabei bezieht sich das „Free“ allein auf die freie Verwendung und Benutzung der Software und nicht auf den Preis im Sinne von kostenlos. Aufgrund vieler missverständlicher Auffassungen des Begriffs hat Richard Stallman einen gelungenen Vergleich veröffentlicht: „To understand the concept, you should think of ‚free‘ as in ‚free speech‘, not as in ‚free beer‘“ [Sta16a]. Unter Freiheit im Sinne der Definition kann also die Freiheit verstanden werden, ein Programm auf dem Computer zu installieren und verwenden, es zu kopieren und verbreiten, die Funktionsweise zu studieren, zu verändern und verbessern.
Die vier Freiheiten lauten in kompakter Form:

1. Die Freiheit, ein Computerprogramm für jeden Zweck ausführen zu können.
2. Die Freiheit, den Quellcode des Computerprogramms auf seine Funktionsweise hin zu untersuchen und diesen auf seine eigenen Bedürfnisse anpassen zu können. Der Zugang zum Quellcode ist dafür eine zwingende Voraussetzung.
3. Die Freiheit, das Computerprogramm zu verbreiten.

4. Die Freiheit, das Computerprogramm zu verbessern und diese Verbesserungen zu veröffentlichen, sodass auch Mitmenschen von der Verbesserung profitieren können. Der Zugang zum Quellcode ist dafür eine zwingende Voraussetzung.

Erst wenn alle vier Freiheiten in einer Lizenz erfüllt sind, ist diese Lizenz „frei“ gemäß der Free Software-Definition. [Sta16a, Teu07]

3.2 Open Source-Definition

Die Open Source-Definition (OSD) wurde von Bruce Perens, der auch die „Debian Free Software Guidelines“ geschrieben hat, ausgearbeitet. Die Definition der Open Source ist deutlich umfangreicher als die der Free Software und zielt dabei auf die gleichen Befugnisse ab. So nehmen auch bei Open Source, neben der Offenlegung des Quellcodes, die Befugnis zur Bearbeitung und Weiterverbreitung eine große Rolle ein.
Insgesamt umfasst die OSD zehn Punkte, die aus der offiziellen Definition der OSI [Ope07a] entnommen wurden. Mit Hilfe der kommentierten Fassung der OSD [Ope07b] sowie der von Christian Teupen [Teu07, S. 59 f.] getroffenen Anmerkungen wurden einzelne Passagen zum besseren Verständnis detaillierter erläutert.

1. **Freie Weiterverarbeitung**
   Die Bedingungen einer Lizenz dürfen keine Einschränkung oder Verbote bzgl. der Weiterverbreitung enthalten. Ferner ist es nicht gestattet, für die Lizenz Gebühren zu verlangen.

2. **Offener Quellcode**
   Die Software muss im Quellcode dauerhaft zugänglich sein. Die Lizenz darf keine Einschränkung oder Verbot enthalten, den Quellcode, weder in kompilierter Form noch im Quellcodeformat, öffentlich zu verbreiten. Für den Fall, dass eine Software ohne Quellcode verbreitet wird, muss dieser auf einfachem Wege offen zugänglich sein (z. B. über das Internet). Eine absichtlich eingefügte Ungenauigkeit in den Quellcode ist nicht gestattet.

3. **Abgeleitete Programme**
   Die Lizenz muss dem Benutzer erlauben, den Quellcode zu modifizieren und aus dessen Grundlage ein neues (abgeleitetes) Programm zu verbreiten. Eine Pflicht, die Modifizierung oder neu entstandene Software unter der gleichen Lizenz zu veröffentlichen, besteht nicht.[1]

4. **Integritätsschutz des originalen Quellcodes**
   Die Lizenz muss das Verteilen von Software, auch wenn es nur einzelne Patchdateien („patch files“) sind, erlauben. Explizit wird mit der zulässigen Verteilung von Patches der Autor geschützt, da dadurch der originale Quellcode von Modifizierungen getrennt wird. Die Lizenz kann darüber hinaus sogar verlangen, dass die aus dem Quellcode

[1] Bei Lizenzen mit Copyleft besteht diese Pflicht, vgl. Kapitel 4.1

abgeleiteten Programme mit einem anderen Namen oder einer anderen Versionsnummer gekennzeichnet werden.

5. **Keine Diskriminierung**
   Eine Open Source-Lizenz darf keine Diskriminierung von bestimmten Personen oder Gruppen enthalten.

6. **Keine Beschränkung der Anwender**
   Die Lizenz darf den Benutzer nicht einschränken, die Software nur in einem bestimmten Bereich einsetzen zu dürfen. Somit ist untersagt, die Software von bestimmten Gebieten auszuschließen (z. B. Gentechnik, Waffenforschung) oder den kommerziellen Einsatz generell zu verbieten.

7. **Verbreitung der Lizenzbestimmungen**
   Die Lizenzbedingungen, die mit dem Programm verteilt werden, gelten für alle verbindlich, ohne dass eine zusätzliche Lizenzvereinbarung getroffen werden muss.

8. **Keine Beschränkung auf ein Produkt**
   Die Lizenzbestimmungen müssen unabhängig davon sein, ob das Programm lediglich ein Teil einer Softwaredistribution ist.

9. **Keine Einschränkung anderer Software**
   Die Lizenz darf keine Einschränkung bzgl. der gemeinsamen Verbreitung von Software beinhalten. Diese Bedingung soll dem entgegenwirken, dass die Software ausschließlich mit anderer Open Source Software weiterverbreitet wird. Außerdem kann in den Lizenzbedingungen nicht verlangt werden, dass Software, die zusammen mit Open Source Software verbreitet wird, auch automatisch als Open Source angesehen wird.

10. **Vertrieb muss technologie-neutral sein**
    Die Lizenz muss so ausgelegt sein, dass sie die verschiedenen Vertriebswege (Internet, CD, usw.) nicht einschränkt.

3.3 Kritischer Vergleich

Inhaltlich unterscheiden sich die beiden Definitionen nicht. Das Ziel beider Initiativen ist es, die Rechte des Endanwenders durch die definierten Freiheiten auszuweiten, da diese Freiheiten bei proprietären Programmen sehr beschränkt sind. Ein Grund warum die OSI ins Leben gerufen wurde, ist die des Marketings. So kann laut den Gründern der OSI der Begriff „Open Source“ auch vom Softwareprozess unabhängigen Personen, wie z. B. dem Unternehmensvorstand oder den Mitgliedern der Aktionärsversammlungen, schmackhaft gemacht werden. Jeder, der den Begriff hört, weiß sofort (anders als bei Free Software), dass der Quellcode Anderen öffentlich zugänglich gemacht wird. Allerdings setzt hier die Kritik von Free Software an, die Stallman in

dem Artikel „Why Open Source misses the point of Free Software“ ([Sta16b]) zusammenfasst und in dem er sich für die Verwendung des Begriffs „Free Software“ anstelle von „Open Source“ einsetzt. Der Hauptkritikpunkt ist, dass zwar mit dem Begriff „Open Source“ deutlich wird, dass der Quellcode offen zugänglich ist, jedoch lässt der Begriff keine Rückschlüsse auf die weiteren Lizenzbedingungen zu. „Open Source ist eine Entwicklungsmethodik; Freie Software ist eine soziale Bewegung“ [Sta16b] ist eine häufig zitierte Aussage Stallmans. Der größte Unterschied zwischen den beiden Initiativen ist nach Stallman, dass das Ziel der OSI einzig und allein darin besteht, Software im praktischen Sinn gemeinsam zu verbessern. Für die FSF ist Freie Software „eine ethisch unbedingt erforderliche, wesentliche Achtung vor der Freiheit der Nutzer“ [Sta16b]. Aus diesem Grund ist die OSI in dieser Hinsicht etwas liberaler und vertritt die Meinung, dass proprietäre und freie Software koexistieren dürfe, während für die FSF nur freie Software existieren solle. Dabei ist wichtig zu erwähnen, dass die FSF ihren „Feind“ nicht im Open Source-Lager sieht, sondern bei proprietärer Software. [J.T01]
In der Praxis wird nicht wie von Stallman gewünscht zwischen Free Software und Open Source Software unterschieden. Es ist sogar geläufig, Open Source als Überbegriff zu verwenden. Um sich dennoch neutral zwischen Free Software und Open Source Software verhalten zu können, werden die Begriffe „Free Open Source Software (FOSS)“ und „Free Libre Open Source Software (FLOSS)“ verwendet. [Sta16b]

Kapitel 4

Open Source-Lizenzmodelle

Die geläufigste Einordnung von Softwarelizenzen ist die über die Stärke des Copylefts. Damit diese Kategorisierung vorgenommen werden kann, wird im Folgenden der Begriff „Copyleft“ und die daraus entstehenden Pflichten erläutert, um anhand dieser Erläuterung bekannte Lizenzen einteilen zu können.
Weiterhin wird in diesem Kapitel mit der Creative Commons eine Non-Profit-Organisation genannt, deren Lizenzen dafür verwendet werden, Werke abseits des Quellcodes lizenzieren zu können. Abschließend ermöglichen die Vor- und Nachteile von Open Source in Bezug zu proprietärer Software einen wichtigen Vergleich und Beitrag zur Entscheidungsfindung, für den Fall freie Software einsetzen oder entwickeln zu wollen.

4.1 Der Begriff „Copyleft“

Damit die Idee der Freien Software nicht in Gefahr läuft, durch kommerzielle Projekte ausgenutzt zu werden, bediente sich die FSF an der gegenteiligen Intention des bekannten Begriffs „Copyright“. Wird eine Software, die unter Copyleft steht, abgeändert und verbessert, müssen diese Modifikationen wiederum auch unter den gleichen Bedingungen wie die Originalsoftware zur Verfügung gestellt werden. So stellte Stallman sicher, dass sich Freie Software weiter verbreitet und daraus keine proprietäre Software entstehen kann.
Das Institut ifrOSS (Institut für Rechtsfragen der Freien und Open Source Software) [Ins16] unterteilt freie Softwarelizenzen nach der Intensität des Copyleft-Effektes. Diese Einstufung wurde für diese Ausarbeitung übernommen. Eine weitere mögliche Einteilung wäre die in BSD-artige (Berkeley Software Distribution), GPL-artige (General Public License) und MPL-artige (Mozilla Public License) Lizenzen.

4.1.1 Lizenzen ohne Copyleft-Effekt

Obwohl das Ziel der FSF war, dass Freie Software frei bleibt, haben sich auch Open Source-Lizenzen durchgesetzt, die nicht mehr mit der einstigen Idee der Freien-Software-Bewegung

(siehe Kapitel 2.1) übereinstimmen. Bei Lizenzen ohne Copyleft-Effekt fällt die zusätzliche Pflicht weg, Anpassungen, Verbesserungen oder Veränderungen an der Software unter der ursprünglichen Lizenz zu veröffentlichen. Somit besteht die Möglichkeit, die einst unter einer Open Source-Lizenz veröffentlichte Software durch Modifikationen in proprietäre Software zu überführen. [Teu07]
In diese Kategorie können alle BSD-artigen Lizenzen, wie z. B. die Apache License eingeordnet werden. Eine Bedingung der BSD-artigen Lizenzen ist, dass der Copyright-Vermerk der originalen Software nicht entfernt werden darf [Ope16]. In der geschichtlichen Entwicklung der FLOSS wurde bereits auf das Mitwirken der BSD bei der Koordination von Unix eingegangen (siehe Kapitel 2).

4.1.2 Lizenzen mit strengem Copyleft-Effekt

Stallmans Intention bei der Gründung der FSF (siehe Kapitel 2.1) war zeitgleich eine Lizenz zu entwickeln, die die vier Freiheiten aus Kapitel 3.1 einhält und bei jedem Projekt verwendet werden kann. Daraus entstand im Jahre 1989 die GNU General Public License (GPL). Dabei war die GPL die erste Lizenz, die vom Copyleft Gebrauch machte. Das heißt, dass auch Veränderungen und Verbesserungen unter der ursprünglichen (freien) Lizenz veröffentlicht werden müssen und somit für jedermann frei zugänglich sind. [Teu07]

4.1.3 Lizenzen mit beschränktem Copyleft-Effekt

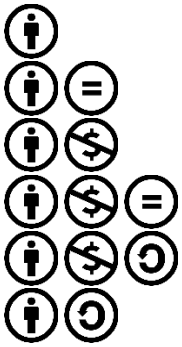
Um auch eine Kombination von unterschiedlichen Lizenzen zuzulassen, gibt es auch Lizenzen mit beschränkten Copyleft. Sie haben den gleichen Effekt wie die Lizenzen aus Kapitel 4.1.2, allerdings können Modifikationen der Software auch unter einer eigenen Lizenz gestellt werden, wie z. B. eine proprietäre Lizenz. Bekanntestes Beispiel ist die MPL[1]. Diese Lizenz verlangt lediglich eine Veröffentlichung für die Modifikation bestehender Dateien. Neu hinzugefügte Dateien können hingegen beliebig lizenziert werden.
Ein weiteres Beispiel ist die abgeschwächte Form der GPL. Die Lesser General Public License (LGPL) wurde speziell für Programmbibliotheken veröffentlicht. Dabei müssen Änderungen an der Programmbibliothek wieder unter der LGPL lizenziert werden (entspricht Copyleft), während das Softwareprogramm, das diese Bibliothek enthält, beliebig lizenziert werden darf (kein Copyleft).

4.2 Creative Commons

Die genannten Softwarelizenzen, die nach Güte ihres Copyleft-Effektes eingeordnet wurden, sind nicht immer die beste Wahl, sein Werk zu schützen. Um auch andere Inhalte zu lizenzieren

[1] Geschichtliche Entwicklung der MPL siehe Kapitel 2.2

wurde einst die GNU Free Documentation License (GFDL) veröffentlicht, welche aber nicht weit verbreitet ist [Sta99] und nur einen Teil dessen abdeckt, für die die Creative Commons große Beliebtheit errang.
Die Creative Commons ist keine Lizenz für Quellcode, sondern für alle anderen Werke. Diese Werke sind vom Urheberrecht geschützt, das die Freiheiten stark einschränkt[2]. Bevor die Creative Commons Lizenzen eingesetzt wurden, hatten die Urheber von Inhalten nur die Wahl sie entweder unter dem gesetzlichen Standardschutz zu stellen („alle Rechte vorbehalten“) oder sie überhaupt nicht zu schützen bzw. zu veröffentlichen. Ersteres ist immer der Fall, wenn der Urheber keine Aussagen bei der Veröffentlichung seines Werkes macht. Die GPL fällt als Lizenz weg, da diese (meist) nur für Quellcode Sinn macht. So müsste ein Musiker bei einer Veröffentlichung eines Musikstückes unter der GPL neben dem eigentlichen Werk auch seine Samples, die MIDI-Dateien und vieles mehr mitliefern, um eine Bearbeitung zuzulassen [Alb12].
Für fast alle Urheber ist es durch die fehlende Expertise im Bereich des Urheberrechts schwierig,



Namensnennung 4.0 international (BY)

Namensnennung-KeineBearbeitung 4.0 international (BY-ND)

Namensnennung-NichtKommerziell 4.0 international (BY-NC)

Namensnennung-NichtKommerziell-KeineBearbeitung 4.0 international (BY-NC-ND)

Namensnennung-NichtKommerziell-Weitergabe unter gleichen Bedingungen 4.0 international (BY-NC-SA)

Namensnennung-Weitergabe unter gleichen Bedingungen 4.0 international (BY-SA)

Abbildung 4.1 Alle möglichen Kombinationen für Lizenzen der Creative Commons und deren Abkürzung (in Anlehnung an [Cre16])

eine geeignete Lizenz für ihren Zweck zu entwerfen. Dieses Problem wurde durch die Verbreitung des Internets verstärkt, da dadurch vermehrt Künstler Zugang zu einem breiten Publikum haben, die sich für die Werke interessieren. Die Lizenzen der Creative Commons helfen dabei nicht nur die Werke passend zu schützen. Nutzer, die das Werk verwenden möchten, sind nun in der Lage, durch die Verwendung der Icons, schnell ihre Rechte abzulesen. Früher wussten sie häufig nicht ob das Werk unter einer Lizenz steht bzw. ohne die meist langen Lizenzbedingungen zu lesen, ob sie den Inhalt weiterverbreiten, bearbeiten oder verwenden dürfen.
Die Creative Commons ist für sich stehend noch keine Lizenz. Die Non-Profit-Organisation bietet sechs vorgefertigte Lizenzverträge, die zum Teil miteinander kombiniert werden können, um rechtliche Bedingungen festzulegen und dem Benutzer zusätzliche Freiheiten zu gewährleisten. Eine Creative Commons Lizenz gewährt somit immer mehr Freiheiten, als das durch den gesetzlichen Standardschutz möglich gewesen wäre.

[2] vgl. § 14 UrhG, § 23 UrhG

Ende 2013 wurde die derzeit aktuellste Version 4.0 der Lizenzen veröffentlicht (siehe Abbildung 4.1). Diese wurden noch nicht ins Deutsche übersetzt, kann dennoch für deutschsprachige Werke eingesetzt werden. [Cre16]

4.3 Abgrenzung zur proprietären Software

Für den Anwender der Software ist der größte Unterschied in Bezug zu proprietärer Software, dass Open Source Software kostenlos oder gegen eine geringe Kopier- oder Supportgebühr zu erhalten ist. Nicht nur deswegen sind Open Source-Produkte bei Privatanwendern und Unternehmern sehr beliebt. Für nahezu alle Aufgabengebiete gibt es das entsprechende Äquivalent in der Open Source-Welt. Allerdings konnten sich häufig die Open Source-Programme nicht gegen die kommerzielle Software durchsetzen. Wie die Statistik in Abbildung 4.2 zeigt, benutzt die

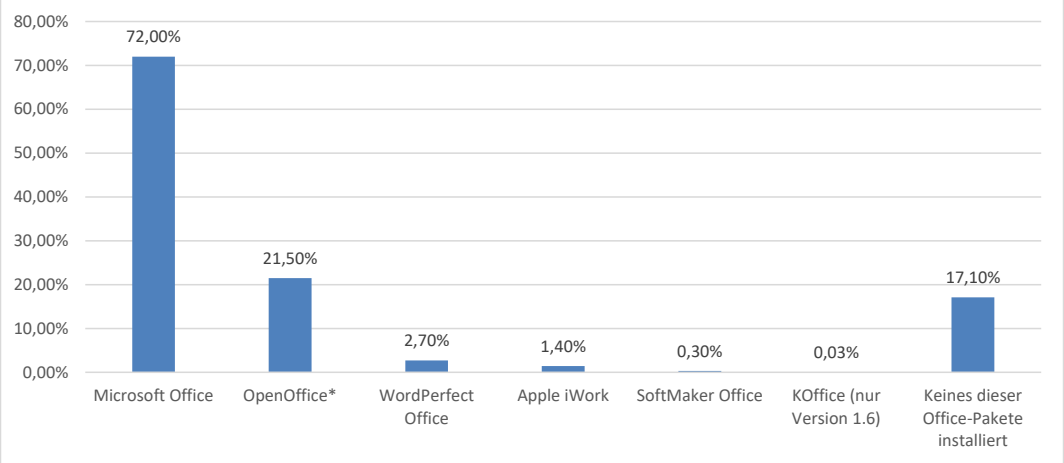


Abbildung 4.2 Verbreitung der Office Programme in Deutschland, Januar 2010 (Mehrfachnennungen möglich) [Web10] *inklusive StarOffice, IBM Lotus Symphony und anderer Derivate.

deutliche Mehrheit bei mehr als einer Million deutschsprachiger Internetnutzer die proprietäre Software Microsoft Office. Hingegen verwenden knapp 22 Prozent das freie Programm OpenOffice (bzw. LibreOffice und andere Derivate). Der Studie zufolge ist der Abstand zwischen den beiden Projekten geringer als das Diagramm zeigt. Bei vielen PCs mit dem Betriebssystem Windows wird damals wie auch heute eine Testversion von Microsoft Office mitgeliefert. Obwohl ein Teil dieser Testversionen bereits abgelaufen ist, und die Nutzer auf andere Office Programme umgestiegen sind, stimmten diese auch für Microsoft Office. Allerdings muss man auch betonen, dass die Studie aus dem Jahr 2010 ist. So wurde Mitte letzten Jahres die neueste Microsoft Office Version (Office 2016) für Mac OS (OS X) veröffentlicht. Außerdem konnte der Absatz von Smartphones und Tablets in den vergangen Jahren angekurbelt werden. Eine weitere Änderung der prozentualen Anteile unter Berücksichtigung dieser mobilen Geräte wird voraussichtlich die Folge sein.
Aus unternehmerischer Sicht mit dem Ziel der Gewinnmaximierung lohnt sich die Entwicklung

Tabelle 4.1 Vor- und Nachteile von Open Source Software

Vorteile	Nachteile
• keine bzw. geringere Kosten	• keine Möglichkeit zur Erhebung von Lizenzgebühren
• höhere Stabilität und Sicherheit	• kleiner Kundenkreis für zus. Dienstleistungen
• freie Anpassung der Software	• durch Offenlegung des Quellcodes womöglich viel Konkurrenz

von Open Source Software zunächst nicht. Durch die fehlende Möglichkeit Lizenzgebühren für die Nutzung zu verlangen, sind Unternehmen im Open Source-Bereich gezwungen, zusätzliche kostenpflichtige Dienstleistungen anzubieten. Selbst dann ist nicht sichergestellt, ob sich diese Vertriebsform langfristig lohnt, da nicht sichergestellt ist, ob die Kunden diese Dienstleistungen nachfragen. Darüber hinaus können durch die Offenlegung des Quellcodes auch andere Unternehmen die Dienstleistungen anbieten. Daraus ergibt sich ein gewisses wirtschaftliches Risiko, falls die Nachfrage der Kunden für diese Dienstleistungen ausbleibt. [Teu07]
Bei der regelmäßigen Nutzung von Open Source Software wird ersichtlich, dass häufiger verbesserte Versionen veröffentlicht werden als dies bei proprietärer Software der Fall ist. Meist liegt das daran, dass bei kommerziellen Produkten dem Kunden nicht zumutbar ist, erneut ein Entgelt für die verbesserte Version zu entrichten[3]. Bei Open Source-Programmen ist die Sachlage eine andere. Dort sind die Entwickler gleichzeitig auch Benutzer des Programms und können damit selbst fehlende Funktionen implementieren und veröffentlichen. Von größeren Open Source-Projekten (wie OpenOffice) sind deswegen unzählige Versionen im Umlauf, wodurch der Laie und sogar involvierte Entwickler schnell den Überblick verlieren können. Im Vergleich zu kommerziellen Produkten besteht diese „Unordnung“ aufgrund eines regulierten Entwicklungsprozesses nicht. [Sch03]
Davon abgesehen gelten Open Source-Programme durch die kurzen Entwicklungszyklen als stabiler und sicherer. Zwar ist der Quellcode offen zugänglich, sodass auch ein möglicher Angreifer Schwachstellen aufspüren und ausnutzen kann. Allerdings kommt es dazu meist nicht, da durch die vernetzte Art der Softwareentwicklung Änderungen am Quellcode automatisch einer Kontrolle ausgesetzt sind. Jedoch sind kommerzielle Produkte auch mehr Attacken ausgesetzt, da Angreifer mögliche Schwachstellen von Kaufsoftware aufzeigen wollen. [Teu07]

[3] Von kleineren Updates, sog. patches, abgesehen

Kapitel 5

Lizenzkonflikte

Durch die vielen nützlichen Vorteile von freiem Code, stellt sich die Frage, ob dieser in einem unter proprietärer Lizenz unterstelltem Quellcode integriert werden darf. Die strenge GPL verbietet eine solche Kombination. Die LGPL bietet eine solche Möglichkeit für Bibliotheken und andere Programme. Auch die liberale BSD-Lizenz erlaubt eine Integration in unfreien Code, wenn der Urheber namentlich genannt wird. Aufgrund der Vielzahl von Lizenzen und der Attraktivität für andere Entwickler, Teile des freien Quellcodes in die eigene Software zu übernehmen, gab es in der Vergangenheit viele Lizenzkonflikte. Interessant ist, dass auch große namhafte Firmen unerlaubt von freiem Code Gebrauch machten.
Vereinfacht dargestellt liegt ein Lizenzverstoß immer dann vor, wenn Quellcode unter einer Lizenz mit (strengem) Copyleft (z. B. die GPL) in Quellcode unter einer nicht Copyleft-Lizenz (z. B. BSD, MPL, Public Domain) integriert wird. In Deutschland kümmert sich die Internetseite gpl-violations.org von Harald Welte um ebensolche Verstöße. Die folgenden zwei Beispiele zeigen die Auswirkungen von Lizenzverletzungen im speziellen Fall der freien Software.

5.1 TomTom Go

Der niederländische Hersteller für Navigationsgeräte „TomTom“ geriet im Jahr 2004 mit seinem PDA-ähnlichen Navigationsgerät „TomTom Go“ in Schwierigkeiten. Damals war TomTom noch ein sehr junges Unternehmen, der das neue Navigationsgerät zu dieser Zeit bereits an etwa 200.000 Kunden ausgeliefert hatte.
Durch die vagen Vermutungen von Christian Daniel und Thomas Kleffel ([DK05]), dass das System auf Linux aufbaut, durchsuchten sie das System nach Beweisen. Schließlich entdeckten sie in der mitgelieferten SD-Karte einen Ordner „system“, in dem sie einen Linux-Kernel in der Version 2.4 extrahieren konnten, der bekanntlich unter der GPL lizenziert ist. Aus diesem Grund meldeten Sie ihre Befunde der Seite gpl-violations.org. TomTom reagierte auf Druck von Harald Welte – auch um ein Gerichtsverfahren zu umgehen – sehr schnell. Sie veröffentlichten auf ihrer

Internetseite[1] den Linux-Kernel samt eigenen Anpassungen. Private Entwickler waren nun in der Lage, mit Hilfe des offenen Quellcodes, das Navigationsgerät mit eigenen Modifikationen zu versehen. [Ble04]
Es konnten keine Informationen gefunden werden, ob und inwiefern auch andere Unternehmen von der Offenlegung profitierten.

5.2 Skype

Skype hat sich im Gegensatz zu TomTom bei seinem Lizenzverstoß nicht vergleichbar vorbildlich verhalten. Das im Jahr 2007 noch Luxemburger Unternehmen[2] wurde wegen eines GPL-Verstoßes vom Landgericht München verklagt. Grund war, dass Skype, der damals VOIP-Software vertrieb, über die eigene Internetseite ein Telefon des spanischen Herstellers „SMC Networks“ anbot. Das auf Linux basierende Telefon lieferte SMC allerdings ohne Quellcode und GPL-Lizenztext aus. Auch in diesem Fall versuchte es Harald Welte zunächst ohne Klage und wies Skype auf die Verletzung der Lizenzbedingung hin. Nachdem SMC darauf nicht reagierte, folgte an Skype eine Abmahnung mit Fristsetzung, den Vertrieb der Telefone einzustellen. Auch nach Ablauf der Frist wurden die Telefone noch angeboten. Allerdings legte SMC dem Telefon ein Beiblatt bei, das auf die Nutzung von GPL lizenzierter Software deutlich macht und die URLs nannte, mit denen der Quellcode heruntergeladen werden kann. Damit wurde allen Anschein nach, die Definition der Freien Software (vgl. Kapitel 3.1) eingehalten, da der Quellcode nun öffentlich zugänglich ist. Das Landgericht München sah dies als nicht ausreichend an. Das Bereitstellen des Quellcodes genügt nur bei Software, die lediglich über das Internet vertrieben werden. Die damalige Klage war an den Anbieter und nicht an den Hersteller des Telefons gerichtet gewesen, da Skype nach Kenntnis der Rechtsverletzung den rechtskonformen Vertrieb des Produkts hätte überprüfen und ggf. einstellen müssen.
Das Urteil, dessen Berufung von Skype durch das Oberlandesgericht München wegen Mangel an Beweisen abgewiesen und somit rechtskräftig wurde, ist vor allem wegen zwei Aspekte sehr interessant. Zum einen wurde mit Skype ein ausländisches Unternehmen von einem deutschen Gericht verurteilt, zum anderen hat das Urteil die Lizenzbedingungen der GPL bestätigt und zeigt auf, dass diese exakt einzuhalten sind. [Sti08, Die07]

[1] Siehe http://www.tomtom.com/en_gb/opensource/
[2] Im Jahr 2011 übernahm Microsoft das Unternehmen Skype

Kapitel 6

Zusammenfassung

Im Fokus dieser Ausarbeitung standen der Vergleich der Freien Software mit der Open Source Software. Wie die geschichtlichen Zusammenhänge am Anfang gezeigt hatten, entstand erst aus der Idee und dem Engagement der FSF letztendlich die OSI, die bis dato als Überbegriff verwendet wird. Hingegen bedienen sich nur fachkundige Personen an neutralen Begriffen wie FLOSS. Bedeutende inhaltliche Unterschiede der beiden Initiativen konnten nicht festgestellt werden. Allerdings vermitteln beide Begriffe eine unterschiedliche Vorstellung. So ist es für die Freie Software eine ethische Herausforderung, quelloffene Software zu verbreiten. Hingegen ist Open Source eine Entwicklungsmethode, bei der freie Software und Kaufsoftware zusammen existieren darf. Allerdings sollten sich Unternehmen genau überlegen, ob und in welcher Form sie ihre Software offen zur Verfügung stellen. Die in Kapitel 4.3 genannten Nachteile können allesamt negative Auswirkungen auf die Gewinnorientierung von Unternehmen haben, während die Vorteile für Privatleute und Unternehmen gemeinsam gelten.
Mit den Lizenzen der Creative Commons wurden Möglichkeiten geschaffen, um andere Werke abseits von Quellcode zu lizenzieren. Durch den strukturierten und einfachen Aufbau der Lizenzen kann hinsichtlich der Lizenzbedingungen auch für Nichtjuristen eine klare Aussage getroffen werden.
Mit den beiden Beispielen für Lizenzverstöße in Kapitel 5 wurde deutlich, dass Unternehmen aus Unwissenheit oder mit Absicht offenen Quellcode für ihre eigenen geschlossenen (meist kommerziellen) Projekte nutzen. Mit dem Gerichtsurteil des Landgericht München aus Kapitel 5.2 wurde ein Exempel statuiert, mit dem Ergebnis, dass die Lizenzbedingungen der GPL in Deutschland exakt einzuhalten sind.

Literaturverzeichnis

[Alb12] ALBRECHT, Robert M.: *Software Lizenzen*. https://romal.de/wp-content/uploads/2015/11/GISWLizenzen.pdf. Version: 20.07.2012

[Ble04] BLEICH, Holger ; HEISE ONLINE (Hrsg.): *Linux-Kernel für TomTom-Go-Navigator veröffentlicht*. https://www.heise.de/newsticker/meldung/Linux-Kernel-fuer-TomTom-Go-Navigator-veroeffentlicht-110692.html. Version: 2004

[Cre16] CREATIVE COMMONS (Hrsg.): *Was ist CC?* http://de.creativecommons.org/was-ist-cc/. Version: 2016

[Die07] DIEDRICH, Oliver ; HEISE ONLINE (Hrsg.): *Urteil gegen Skype wegen GPL-Verletzung*. https://www.heise.de/newsticker/meldung/Urteil-gegen-Skype-wegen-GPL-Verletzung-154544.html. Version: 2007

[DK05] DANIEL, Christian ; KLEFFEL, Thomas: *Hacking TomTom GO*. https://events.ccc.de/congress/2005/fahrplan/attachments/569-Paper_HackingTomTomGo.pdf. Version: 2005

[Gra04] GRASSMUCK, Volker: *Schriftenreihe / Bundeszentrale für Politische Bildung*. Bd. 458: *Freie Software: Zwischen Privat- und Gemeineigentum*. [Online-Ausg.], 2., korr. Aufl., Red.-Schluss: 30. November 2004. Bonn : Bundeszentrale für Politische Bildung, 2004 http://freie-software.bpb.de/Grassmuck.pdf. – ISBN 3–89331–569–1

[Ins16] *Lizenz-Center | ifrOSS*. http://www.ifross.org/lizenz-center. Version: 2016

[J.T01] J.T.S. MOORE ; J.T.S. MOORE (Hrsg.): *Revolution OS*. https://www.youtube.com/watch?v=4vW62KqKJ5A. Version: 2001

[Kre16] KREMPL, Stefan ; HEISE ONLINE (Hrsg.): *Linux in München: Berater empfehlen Ausstieg aus LiMux auf Raten*. https://www.heise.de/newsticker/meldung/Linux-in-Muenchen-Berater-empfehlen-Ausstieg-aus-LiMux-auf-Raten-3463100.html. Version: 2016

[Lan11] LANDESHAUPTSTADT MÜNCHEN (Hrsg.): *Das LiMux-Projekt der Landeshauptstadt München: Der Weg aus der Herstellerabhängigkeit zu Open Source.* http://www-05.ibm.com/de/services/referenzen/downloads/cs-stadt-muenchen-op.pdf. Version: 2011

[Ope07a] OPEN SOURCE INITIATIVE (Hrsg.): *The Open Source Definition*. https://opensource.org/osd. Version: 2007

[Ope07b] OPEN SOURCE INITIATIVE (Hrsg.): *The Open Source Definition (Annotated)*. https://opensource.org/osd-annotated. Version: 2007

[Ope16] *The 2-Clause BSD License | Open Source Initiative*. https://opensource.org/licenses/bsd-license.php. Version: 2016

[Ray99] RAYMOND, Eric S.: *The cathedral and the bazaar: Musings on Linux and Open Source by an accidental revolutionary*. 1. ed. Beijing : O'Reilly, 1999. – ISBN 1565927249

[Sch03] SCHIFFNER, Thomas: *Rechtswissenschaftliche Forschung und Entwicklung*. Bd. 690: *Open Source Software: Freie Software im deutschen Urheber- und Vertragsrecht: Zugl.: München, Univ., Diss., 2002*. München : VVF, 2003. – ISBN 3894814659

[Sta99] STALLMAN, Richard: *GNU Free Documentation License Version 0.9 DRAFT*. https://groups.google.com/forum/#!topic/gnu.misc.discuss/xsRJ5cUIR_g. Version: 1999

[Sta16a] STALLMAN, Richard ; FREE SOFTWARE FOUNDATION, Inc. (Hrsg.): *What is free software? The Free Software Definition*. https://www.gnu.org/philosophy/free-sw.en.html. Version: 2016

[Sta16b] STALLMAN, Richard: *Why Open Source misses the point of Free Software*. https://www.gnu.org/philosophy/open-source-misses-the-point.en.html. Version: 2016

[Sti08] STIEBERT, Julia ; GOLEM.DE (Hrsg.): *Skype zieht Einspruch im GPL-Verfahren zurück: Gericht bestätigt Lizenz für freie Software erneut*. http://www.golem.de/0805/59587.html. Version: 2008

[Teu07] TEUPEN, Christian: *Copyleft" im deutschen Urheberrecht: Implikationen von Open Source Software (OSS) im Urhebergesetz*. Duncker & Humblot, 2007 (Schriften zum Bürgerlichen Recht). – ISBN 9783428123254

[Web10] *Verbreitung von Office-Software bei Internetnutzern in Deutschland im Januar 2010*. http://www.webmasterpro.de/portal/news/2010/01/25/verbreitung-von-office-programmen-openoffice-ueber-21.html. Version: 2010