FACHHOCHSCHULE WEDEL

S E M I N A R A R B E I T

In der Fachrichtung
Wirtschaftsingenieurswesen
SS 2018

Thema Z8

Open Source Software Vs. Proprietärer Software

Eingereicht von:	Olalekan Ale (Mat.-Nr. 101816)
	E-Mail: wing101816@fh-wedel.de
Erarbeitet im:	6. Semester
Abgegeben am:	18.05.2018
Betreuer (FH Wedel):	Prof. Dr. Michael Anders Fachhochschule Wedel Feldstraße 143 22880 Wedel Tel. 04103 – 80 48 – 35 E-Mail: an@fh-wedel.de

Inhaltsverzeichnis

1. Einleitung

In der Computerindustrie hat Open Source Software (OSS) in den letzten Jahren großen Zuspruch erhalten und ist mittlerweile dank dem Internet so weit verbreitet, dass nahezu jeder (und sei auch nur unbewusst) auf sie zurückgreift. Ging man Anfang der 2000er Jahre noch davon aus, dass sich bewährte proprietäre IT Strukturen in z. B. Bibliotheken nicht durch OSS ersetzen lassen, lässt sich heute ein anderes Bild sehen.[1] Diverse CMS, ILS- Anwendungen, Kataloge, Retrieval Systeme und Cloud Dienste lassen sich mit OSS betreiben. OSS und vor allem sein prominentester Vertreter Linux wurden mit der Zeit zunehmend als hochwertige Alternative anerkannt, von einer Vielzahl von Anwendern auf Rechnern verwendet und bildeten schließlich den Kern neuer gewinnorientierter Strategien und Geschäftsmodelle. Google, Facebook, Amazon und viele weitere Unternehmen basieren auf OSS oder verwenden OSS.[2] Heute ist OSS in vielen Bereichen zum Standard geworden, denn mit dem offen gelegten Quellcode lässt sich die Software flexibel an eigene Bedürfnisse anpassen und läuft in der Regel stabil als auch sicher. Sie wird nicht als proprietärer Software eines Unternehmens entwickelt. Obendrein werden keine Lizenzgebühren verlangt und der Download aus dem Internet ist für jeden kostenlos.

Ziel dieser Arbeit ist es sowohl die Stärken als auch Schwächen von OSS und proprietärer Software zu untersuchen und gegenüberzustellen. Hierbei wird besonders die Sicherheit hervorgehoben. Darüber hinaus wird OSS im kommerziellen Bereich untersucht.

Im 2. Kapitel werden die Grundlagen von OSS behandelt. Es wird zunächst das Konzept „Open Source Software“ beschrieben und von anderen Softwarekategorien wie Proprietärer Software abgegrenzt. Im Genauen werden wichtige Begriffe bzw. Eigenschaften erklärt und der historische Hintergrund sowie einige Open Source Software Lizenzen vorgestellt.

Das 3. Kapitel befasst sich mit der Entwicklung von Open Source Projekten. Wie entsteht trotz eines solch unkonventionellen Entwicklungsmodells qualitative hervorragende Software? Wie ist deren Arbeitsteilung und wie unterscheidet sie sich von proprietär erzeugter Software? Diese Fragen werden diskutiert, denn jedes große Projekt mit einer Vielzahl beteiligter Personen benötigt eine sorgfältige Arbeitsteilung und setzt allgemein akzeptierte Regeln zur Zusammenführung der geleisteten Ergebnisse voraus. Weiterhin wird ein Überblick über einige relevante Open Source Anwendungen gegeben.

Anschließend widmet sich das 4. Kapitel einer groben Gegenüberstellung der Vor- und Nachteile von OSS gegenüber proprietärer Software.

Im 5. Kapitel wird die Sicherheit und Verlässlichkeit von OSS genauer untersucht und mit proprietärer Software verglichen.

Welche Möglichkeiten gibt es, mit OSS Geld zu verdienen? Das 6. Kapitel beschreibt kurz existierende Geschäftsmodelle für OSS.

Zu guter Letzt erfolgt im 7. Kapitel eine Zusammenfassung und Interpretation der Ergebnisse.

[1] (Breeding, 2002)
[2] (Martin-Jung & Neidhart, 2018)

2. Open Source Software und proprietärer Software

In diesem Kapitel erfolgt die Abgrenzung und Definition verschiedener Softwarearten. Dies dient zum Verständnis der weiterführenden Analysen. Was genau ist OSS? Was ist der Unterschied zu Proprietärer Software?
Um diese Fragen zu beantworten wird ein Blick auf die Entstehungsgeschichte der OSS-Bewegung geworfen. Zusätzlich werden die von der Open Source Initiative (OSI) veröffentlichten OSS Kriterien aufgelistet. Abschließend werden einige Wichtige OSS Lizenzen[3] mit ihren Eigenschaften beschrieben und erklärt welch wichtige Bedeutung sie für OSS haben.

2.1 Was ist OSS?

Als Open Source Software bezeichnet man Software, deren Quellcode (engl. source code) im Gegensatz zu proprietärer Software offen zugänglich ist. Der Quellcode bezeichnet die vom Programmierer in der „höheren", menschlichen lesbare Programmiersprache geschriebenen Anweisungen. Diese Anweisungen sind aufgrund Ihrer Syntax und Struktur für andere Programmierer verständlich. Damit die Anweisungen vom Computer durchgeführt werden, muss der Quellcode durch einen Compiler (ein spezielles Programm) in die maschinenlesbare Form (Objektcode) gebracht werden. Weitere wichtige Merkmale die eine OSS definieren sind:[4]

- Jeder hat das recht die Software nach eigenem Ermessen zu nutzen.
- Die Software darf vom Nutzer ergänzt und verändert werden.
- Der Benutzer darf die Software in modifizierter wie in nichtmodifizierter Form lizenzkostenfrei vervielfältigen und verbreiten.

OSS wird auch als „Free Software" (Freie Software) bezeichnet, denn dem Verwender wird eine weitreichende Freiheit in der Nutzung gewährt. Hier bezieht sich „Frei" nicht auf den Preis, sondern auf „die Freiheit in der Nutzung". Der Unterschied zwischen den Begriffen ist, dass OSS eher auf eine pragmatische Sichtweise hindeutet, wohingegen der Begriff der „Free Software" breiter angelegt ist. Er stellt die Freiheit des Nutzers im Umgang mit der Software stärker in den Vordergrund und ist auch philosophisch und gesellschaftspolitisch motiviert.
Der Term „Open Source Software" hat sich heute weitgehend durchgesetzt.
OSS muss nicht zwingend kostenlos sein. Es können z. B. Gebühren für das Kopieren der Software oder den Erwerb angebotener Zusatzleistungen wie Support verlangt werden.[5]

2.2 Was ist proprietäre Software?

Als proprietäre Software bezeichnet man Software deren Quellcode geheim gehalten wird, wodurch sie nicht für alle zugänglich ist. Die Software wird sozusagen als geschütztes Eigentum mit Ausschließlichkeitsrechten behandelt und deren Nutzung wird in eigeschränktem Umfang lizenziert.[6]
Proprietäre Software wird in der Regel kommerziell vertrieben, weshalb häufig auch der Terminus „kommerzielle Software" verwendet wird. Die käuflich zu erwerbende Lizenz räumt das einfache Nutzerrecht ein, verbietet jedoch das Kopieren und verbreiten der Software. So wird die Einnahme weiterer Lizenzgebühren sichergestellt.

Unter proprietärer Software fallen auch die Softwaremodelle Shareware und Freeware. Unternehmen verwenden diese beiden Modelle als strategisches Instrument, um am an Marktbekanntheit zu gewinnen. Freeware ist Software die kostenlos genutzt werden kann. Hier steht „Free" tatsächlich für den Preis. Außer dass die Software kostenlos genutzt werden kann gibt es keinen wesentlichen Unterschied zur herkömmlichen proprietären Software.

---

[3] Weitere Lizenzen sind auf sind auf www.opensource.org zu finden
[4] vgl. www.opensource.org für vollständige Definition.
[5] (Raymond, The Revenge of the Hackers, Voices from the Open Source Revolution, 1999), (Stallman, 2002)
[6] (Jaeger & Metzger , 2002)

Bei Shareware liegt der Unterschied darin, dass die Software in ihrer Funktion oder zeitlichen Nutzung eingeschränkt ist. Beiden Modellen ist das Kopieren und die Weitegabe erlaubt.

2.3 Historischer Hintergrund[7]

Die Geschichte der OSS reicht bis in die sechziger Jahre zurück, denn zu der Zeit war alle Software quelloffen. Software stellte kein Gut mit eigenem wirtschaftlichem Wert dar, sondern die wurde jeweils für eine spezielle Hardware erstellt und mit dieser verkauft. Das Computergeschäft bestand ausschließlich aus dem Verkauf und der Wartung von Hardware.
Hauptsächlich Programmierer und Systementwickler die ihre Anwendungen selbst schrieben und untereinander ausgetauscht haben, haben Computer genutzt. Es bildete sich eine sogenannte „Hacker Kultur“, zu deren Leitbildern die absolute Informationsfreiheit, das Prinzip der dezentralen Machtverteilung sowie der Glaube, dass Computercodes für alle unbegrenzt zugänglich sein sollten, gehörten.[8]

In den siebziger Jahren gewann Software mehr und mehr an Bedeutung, was damit zu tun hatte, dass das 1969 US-Justizministerium ein Kartellverfahren gegen IBM (Marktanteil: ca. 70 Prozent) einleitete. Das Verfahren führte dazu, dass die Bündelung von Soft- und Hardware aufgegeben wurde und sich so ein eigenständiger Softwaremarkt bilden konnte. Die ersten Softwareunternehmen wie z. B. Microsoft (1974) von Bill Gates und Paul Allen entstanden. Bill Gates wehrte sich als einer der ersten Softwareentwickler gegen die Erstellung von Raubkopien.[9]
Zu Beginn der achtziger Jahre wurde fast alle Software proprietär vermarktet. Software wurde ein Massenprodukt, denn auch im privaten Bereich konnte Software ohne Programmierkenntnisse verwendet werden.

1969 wurde von AT&T[10] Bell Laboratories das Betriebssystem Unix entwickelt, welches sehr flexibel und auf den verschiedenen vorhandenen Rechnern eingesetzt werden konnte.
Da es AT&T aus kartellrechtlichen Gründen verboten war, in anderen Bereichen außer der Telekommunikation wirtschaftlich aktiv zu sein, konnte Unix nicht kommerziell vermarktet werden. Daher wurde der Quellcode zum Selbstkostenpreis im Quelltext an Universitäten, Forschungseinrichtungen und andere Unternehmen lizenziert.
Da es keinen offiziellen Support gab, halfen sich Unix Nutzer gegenseitig aus und tauschten ihre Erweiterungen bzw. Veränderungen des Softwarecodes aus. Im Laufe der Zeit entstanden viele Versionen die kommerziell von verschiedenen Unternehmen vertrieben wurden und nicht mehr im vollen Umfang zueinander kompatibel waren. Die University of California in Berkeley diente als Sammelstelle für Erweiterungen Unix Codes, woraufhin sich die Berkeley Software Distribution (BSD) bildete und im Laufe der Jahre eine eigene Unix Variante unter der BSD Lizenz vertrieb.
1984 spaltete sich AT&T auf und damit endete auch dessen Zwangsregulierung. AT&T konnte nun Unix selbst kommerziell verwerten. Sie erhöhten die Lizenzgebühren so stark, dass sich ein Einsatz nur noch für wenige Unternehmen lohnte.

Mit der Gründung des GNU Projekts (1884) und der „Free Software Foundation“ (1995) von Richard Stallmann gelang es, ein Revival Freier Software einzuläuten.
Anlass für diese Entscheidung, war Stallmanns Versuch Zugang zum Quellcode eines Druckertreibers zu erlangen um die Behandlung von Druckfehlern zu verbessern.
Der Druckerhersteller jedoch verweigerte ihm dem Zugang zum Quellcode, da dieser als Betriebsgeheimnis gewahrt wurde. Stallmann hatte das Ziel, der aufkommenden Kommerzialisierung der Softwareindustrie ein freies Softwareuniversum entgegenzusetzen und freie Alternativen zu proprietärer Software zu schaffen.[11] Um dein Ziel zu erreichen gründetet er das GNU Projekt

[7] Dieser Abschnitt basiert auf den Darstellungen von (Moody, 2002), (Grassmuck, 2004), (Torvalds & Diamond , 2001) und (Raymond, A Brief History of Hackerdom, 1999)
[8] (Levy, 2001)
[9] (Gates, 1976)
[10] American Telephone & Telegraph Corporation
[11] (Williams, 2002)

(„GNU`S Not Linux“) um ein Unix-ähnliches Betriebssystem zu entwickeln. Für die Koordination des Ganzen, gründete er die Free Software Foundation (FSF) und entwickelte das GNU General Public License (GNU GPL) mit dem sogenannten „Copyleft“.
Unter dem Gnu Projekt entstanden viele hochwertige Programme. Doch das eigentliche Ziel ein Unix-ähnliches Betriebssystem zu schaffen, wurde erst realisiert als der damalige Student Linus Torvalds in Helsinki unabhängig vom GNU Projekt, das Betriebssystem Kernel „Linux“ unter der GPL veröffentlichte. Zusammen mit den Programmen aus dem GNU Projekt und dem Kernel entstand das noch heute bekannte GNU/ Linux System.

In den neunziger Jahren gewann die Open Source Entwicklung zunehmend an Aufmerksamkeit. Durch die Weiterentwicklung des Internets zum universellen Web, entstanden viele neue Projekte, darunter das Apache Projekt 1995, KDE 1996 und das Mozilla Projekt 1998. Im Rahmen der Freilegung des Quellcodes für den Internet Browser „Communicator“ unter dem Codenamen Mozilla wurde auch der Begriff „Open Source Software“ geprägt. Man habe einen neuen Begriff gesucht, der den missverständlichen Terminus „Free Software“ ersetzen sollte, um quelloffene Software besser vermarkten zu können. So gründeten Eric Steven Raymond und Bruce Perens die Open Source Initiative (OSI). 1999 veröffentlichte Bruce Perens die „Open-Source-Definition[12]“ in der die Kriterien festgehalten wurden, die eine Open Source Software ausmacht:

Die Open-Source-Defnition

- *Freie Weitergabe:* Die Lizenz darf niemanden darin hindern, die Software zu verkaufen oder sie mit anderer Software zusammen in einer Software-Distribution weiterzugeben. Die Lizenz darf keine Lizenzgebühr verlangen.
- *Verfügbarer Quellcode:* Die Software muss im Quellcode für alle Nutzer verfügbar sein.
- *Abgeleitete Arbeiten:* Die Lizenz muss von der Basissoftware abgeleitete Arbeiten und deren Distribution unter derselben Lizenz wie die Basissoftware erlauben.
- *Integrität des Autoren Quellcodes:* Eine Einschränkung der Weitergabe von verändertem Quellcode ist nur dann zulässig, wenn die Weitergabe des Originalcodes zusammen mit „Patch Files“ erlaubt ist, die den Code bei der Kompilierung verändern
- *Keine Diskriminierungen von Personen oder Gruppen:* Die Lizenz darf nicht einzelnen Personen oder Gruppen die Nutzung der Software verweigern.
- *Keine Einschränkung bezüglich des Einsatzfeldes der Software:* Die Lizenz darf den Verwendungszweck der Software nicht einschränken, z. B. kein Ausschluss militärischer oder kommerzieller Nutzung o. ä.
- *Weitergabe der Lizenz:* Die Lizenz muss für alle zutreffen, welche die Software erhalten, ohne z. B. eine Registrierung oder eine andere Lizenz erwerben zu müssen.
- *Die Lizenz darf nicht auf ein bestimmtes Produktpaket beschränkt sein:* Die Lizenz muss produktneutral gestaltet sein und darf sich z. B. nicht auf eine bestimmte Distribution beziehen.
- *Die Lizenz darf die Weitergabe zusammen mit anderer Software nicht einschränken:* Sie darf zum Beispiel nicht verlangen, dass sie nur mit Open Source Software verbreitet werden darf.
- *Die Lizenz muss Technologie-neutral sein:* Sie darf z. B. nicht verlangen, dass die Distribution nur via Web/CD/DVD verteilt werden darf.

[12] In Anlehnung an (Perence , 1999), (Mundhenke, 2007),

Neben der Entwicklung von Open Source Projekten, verwaltet die OSI Softwarelizenzen welche die Kriterien der „Open Source Definition“ einhalten.
Seither hat Open Source Software auch viele kommerzielle Unternehmen als Unterstützer gewinnen können. Früher Befürworter von OSS waren Beispielsweise IBM, Netscape und Oracle.

2.4 Open Source Lizenzen[13]

Die Open Source Initiative (OSI) führt eine Liste von der derzeit über 50 anerkannter Open Source Lizenzen, die mit der Open Source Definition übereinstimmen. Das heißt: Die Lizenz erlaubt im Gegensatz zur proprietären Software die freie Nutzung, Bearbeitung und Weitergabe (unter bestimmten Bedingungen) des Quelltextes. zusätzlich beinhalten die Lizenzen verschiedene Obligationen, Definitionen und teils Absätze zu Patenten, Gerichtsständen und so weiter.[14]

Die Lizenzen lassen sich grob in „Permissive License“ (freigiebige Lizenz) und „Non Permissive License“ (nicht-freigiebige Lizenz) aufteilen. Dies bezieht sich auf den von Richard Stallmann entwickelten „Copyleft Effekt“.
Mit „Copyleft“ ist gemeint, dass die durch die Lizenz gewährten Rechte weitergegeben werden müssen, wenn der so lizenzierte Code für eine neue Software verwendet wird. Unabhängig davon, ob ein eigener Code hinzukommt, das Ergebnis stünde wieder unter der Lizenz der Ausgangs-Software. Somit gewährleistet der „Copyleft Effekt“, dass vom Urheber frei lizenziertes Material nicht ohne Weiteres in proprietäre Produkte verwendet werden können.
Ein bekanntes Beispiel für eine Non Permissive License (nicht-freigiebig) ist die General Public License (GNU GPL). Das beste Beispiel für eine Permisive License (freigiebig) ohne „Copyleft Effekt“ ist die Berkeley Software Distribution License (BSD). Ein Programmierer, der ein unter einer BSD-Lizenz veröffentlichte Software verändert und dann binär verbreitet, ist nicht verpflichtet, den Quellcode mitzuveröffentlichen. Jede Weiterverbreitung und Verwendung in nichtkompilierter oder kompilierter Form, mit oder ohne Veränderung, muss jedoch weiterhin unter BSD-Lizenz erfolgen. Dafür muss dem Programm der BSD-Lizenztext hinzugefügt werden. Aus einer BSD-lizenzierte Software können somit proprietäre Software Derivate entstehen.

Im Folgenden werden einige von der OSI anerkannte Lizenzen aufgeführt:

GNU General Public License (GNU GPL):

Die GNU General Public License (GNU GPL) ist die am weitesten verbreitete Open Source Software Lizenz. Diese verpflichtet den Nutzer dazu, bei Weiterverbreitung der Software in ihrer ursprünglichen oder abgeleiteten Form ebenfalls unter die Vorgaben der GPL zu stellen (sog. „Copyleft Effekt“). Hält sich der Lizenznehmer nicht an die Bedingungen, verliert er die Befugnis zur freien Benutzung rückwirkend. Auf diese Weise wird sichergestellt, dass auch sämtliche abgeleitete Werke Open Source bleiben („viraler Effekt“).

Beispielsweise wurden das Betriebssystem Linux und das Datenbankverwaltungssystem MySQL unter der GNU GPL veröffentlicht.

[13] (Anonym, Licenses & Standards, 2018)
[14] (Lang & Augsten, 2017)

BSD Lizenz (Berkeley Software Distribution-Lizenz)

Die BSD Lizenz, die von der University of California, Berkeley stammt, hat als einzige Bedingung, dass der Copyright-Vermerk der ursprünglichen Software nicht entfernt werden darf. Somit kann unter einer BSD-Lizenz stehende Software auch als Vorlage für proprietäre Software dienen, denn der Lizenznehmer ist bei dieser Lizenz nicht verpflichtet den Quellcode offen zu legen. Wichtig ist, dass die Weiterverbreitung und Verwendung in nichtkompilierter oder kompilierter Form weiterhin unter der BSD-Lizenz erfolgen muss. Dazu muss der proprietären Software der BSD-Lizenztext hinzugefügt werden.

GNU Lesser General Public License (LGPL)

Aufgrund der sehr restriktiven Lizenzpolitik der GNU GPL wurde die LGPL als Spezialfall der GPL für bestimmte Softwarebibliotheken entwickelt. Eine Bibliothek ist die Sammlung von Softwarefunktionen und Daten, die mit Anwendungsprogrammen verbunden werden können, um die Entwicklung von Programmen zu vereinfachen.
Sofern die Bibliothek unverändert genutzt wird, steht nur diese unter der LGPL, die übrigen Softwareteile können anderweitig lizenziert werden. So kann LGPL-Software auch in proprietärer Software genutzt werden, ohne dass der Quellcode der eigenen Software Teile für alle zugänglich sein muss.

3. Open Source Projekte

Das 3. Kapitel befasst sich mit der Entwicklung von Open Source Projekten, d.h. die Art und Weise, wie die Organisation der Open Source Entwicklung aufgebaut ist und abläuft.
Hier wird als grundlegender Unterschied zur Entwicklung proprietärer Software die anscheinende unkoordinierte Zusammenarbeit einer weltweit verstreuten Community genannt, was es umso erstaunlicher macht, dass aus den Beiträgen der einzelnen Entwickler eine hochwertige Computersoftware entsteht. Es wird nur eine sehr starke Vereinfachung betrachtet, die dem echten komplexen Ablauf nicht ganz gerecht wird.
Weiterhin wird ein kurzer Überblick über einige relevante Open Source Anwendungen gegeben.

3.1 Entwicklungsstruktur von Open Source Projekten

Die Entwicklung proprietärer Software findet i. d. R. in einem Unternehmen statt. Dabei wird ein geschlossener, zentralisierter und klar strukturierter Entwicklungsprozess verfolgt.[15]

Im Gegensatz dazu werden OSS Projekte durch Projektmitglieder, die über das Internet miteinander kommunizieren, dezentral entwickelt. Eine genaue hierarchische Beziehung wie bei der Entwicklung proprietärer Software gibt es normaler weise nicht, sondern die informelle Autorität der fähigsten Entwickler gilt. Eine der ersten Arbeiten die sich mit Prinzipien der OSS Entwicklung beschäftigt, ist ein vielbeachteter Aufsatz von Raymond.[16]
Raymond vergleicht die beiden Entwicklungsmodelle als Kathedralen- (klassische Softwareentwicklung) und Basar-Modell (OSS-Entwicklung). Die Kathedrale wird als eine geschlossen hierarchisch organisierte Gruppe von eingeweihten Spezialisten betrachtet, während der Basar als Modell für ein auf den ersten Blick unorganisiertes Durcheinander steht, auf den zweiten Blick aber eine besondere Leistungsfähigkeit aufgrund des starken Austausches der Mitglieder erlangt.
An der Spitze eines Open Source Projekts steht meistens der sogenannte Projektleiter (kann eine einzelne Person oder eine Gruppe sein).

---

[15] (Hoch, Roeding , & Purkert, 1999)
[16] (Raymond, The Cathedral & the Bazaar, 2001)

Oft handelt es sich dabei um den ursprünglichen Initiator des Projekts, der den ersten Softwarecode entwickelt hat. Er gibt die Leitlinien der Projektentwicklung vor und trifft die ultimativen Entscheidungen.[17] Vor allem bei größeren Projekten wird der Projektleiter von einer Gruppe von Peers (auch Maintainer genannt) unterstützt. Ihnen werden einzelne Bereiche zugeteilt und deren Aufgabe ist es Beiträge von anderen Programmierern zu sichten und zu bewerten. Unter den Peers befinden sich die Programmierer die aktiv an der Ergänzung, Fehlerkorrektur und Weiterentwicklung der Quellcodes arbeiten. Ferner gibt es die Bug Fixer, die zwar wenige Entwicklungsbeiträge leisten oder einzelne Fehler melden, dennoch sind sie ein wichtiger Bestandteil damit eine qualitative hochwertige und sichere Software entsteht. Grundlage für die Zusammenarbeit ist das Einhalten offener Standards für Protokolle, Dateiformate und Schnittstellen, welche die Basis für die Entwicklung bilden. Ohne einhalten offener Standards kann es passieren, dass Teilprojekte inkompatibel zueinander entwickelt werden.
Bei proprietärer Softwareentwicklung gibt es eine klare Trennung zwischen Entwicklungsphase und Testphase. Häufig werden Beta Tester vom Unternehmen ausgewählt, die nach Fehlern suchen und diese den Entwicklern melden.
Bei OSS kann jeder Anwender ein Tester sein. Aufgrund des offenen Quellcodes haben alle Personen die Möglichkeit, den Quellcode zu lesen, zu verändern und damit zu Mitentwicklern zu werden.

3.2 Offener Standard[18]

Ein Offene Standard bezieht sich auf ein Format oder Protokoll, das:

1. öffentlich zugänglich ist, zur öffentlichen Bewertung und Verwendung, ohne Einschränkungen und für alle beteiligten Teilnehmer gleichwertig,
2. ohne Bestandteile oder Erweiterungen ist, deren Abhängigkeiten wiederum selbst nicht der Definition eines Offenen Standards entsprechen,
3. frei von rechtlichen oder technischen Bestimmungen ist, die die Verwendung von irgendeinem Beteiligten oder Geschäftsmodell einschränken,
4. unabhängig von einem einzigen Anbieter in einem Prozess weiterentwickelt wird, der offen für eine gleichberechtigte Beteiligung von Wettbewerbern und Drittanbietern ist,
5. verfügbar in mehreren vollständigen Implementierungen ist, entweder von konkurrierenden Anbietern, oder als eine vollständige Implementierung, die gleichberechtigt verfügbar für alle Beteiligte ist.

Auf diese Weise wird sichergestellt, dass Technik für jeden verfügbar ist, unabhängig vom Geschäftsmodell, der Größe oder Beständen an Exklusivrechten.

[17] (Lerner & Tirole, 2000)
[18] Definition laut Free Software Foundation Europe (FSF) (Anonym, Offene Standards, kein Datum)

3.3 Open Source Projekte (Beispiele)

Linux[19]

Linux ist ein plattformunabhängiges Betriebssystem, welches 1991 vom damals 21-Jährigen Studenten Linus Torvalds entwickelt wurde. Genau genommen bezeichnet „Linux“ nur den Betriebssystem-Kernel, der auf dem Minix-Betriebssystem, ein UNIX-artiges Betriebssystem aufbaut. Erst in Verbindung mit den vom GNU-Projekt entwickelten Softwarewerkzeugen (Editor, Compiler und andere Utilitys), liegt das vollständige Betriebssystem „GNU/Linux“ vor. Der Kernel fungiert dabei u. a. als Schnittstelle zu Anwenderprogrammen (Ein-/Ausgabe-Operationen), kontrolliert den Zugriff auf Prozessor, Geräte und Speicher, strukturiert und verteilt Ressourcen (Lastverteilung) und überwacht die Zugriffsrechte auf Dateien und Geräte.[20]
Die Zusammensetzung des Linux-Kernels mit weiterer Hilfssoftware wie z. B. GNU Core Utilities und andere Anwendungsprogramme nennt man „Linux-Distributionen“.
Je nach Bedürfnis können Distributionen unterschiedlich sein. Bekannte Herausgeber von Linux-Distributionen sind z. B. Ubuntu, Knoppix und. Red-Hat.
Linux ist heute für eine Vielzahl von Hardware Architekturen verfügbar und wird in fast allen Bereichen der Computertechnik eingesetzt. Die Einsatzbereiche reichen von Desktop Rechnern und Servern über Mobiltelefone bis hin zu Hochleistungsrechnern.
Linux ist bei vielen Unternehmen sehr hoch angesehen aufgrund der großen Bandbreite seiner Einsatzmöglichkeiten, zumal man durch die verschiedensten Distributionen das Betriebssystem auf die gewünschten Einsatzzwecke anpassen kann.
Einige Hardware Hersteller, wie z. B. Sun und IBM, liefern ihre Maschinen auf Wunsch mit Linux aus.

Apache[21]

Der Apache HTTP Server ist der meistgenutzte Webserver im Internet und wurde 1995 erstmals publiziert. Er war das Ergebnis der Zusammenarbeit einer kleinen Entwicklergruppe, die sich das Ziel gesetzt hatten, einige Bug-Fixes und Patches für eine bereits im National Center for Supercomputing Applications (NCSA), an der Universität von Illinois, eingesetzte Software zusammenzutragen. Der Server ist neben Unix und Linux auch auf Windows sowie einer Vielzahl weiterer Betriebssysteme lauffähig. Zusätzlich ist er modular aufgebaut und bietet durch Verwendung serverseitiger Skriptsprachen, die Möglichkeit, Webseiten dynamisch zu erstellen. Häufig verwendete Skriptsprachen sind PHP oder Perl.[22] Seit 1995 wird die Weiterentwicklung des Webservers durch die Apache Software Foundation koordiniert.

[19] (Anonym, The Linux Kernel Archieves, kein Datum)
[20] (Mundhenke, 2007)
[21] (Anonym, The Apache Software Foundation, kein Datum)
[22] (Mundhenke, 2007)

Weitere nennenswerte Open Source Projekte sind:

Software Kategorie	OSS	Proprietärer Software
Betriebssysteme	Linux Ubuntu	MS Windows
Webserver	Apache	MS IIS
Texteditor	Notepad, OpenOffice, Libreoffice	MS Word
Media Player	VLC, Kodi, Audacity	Windows Media Player
Datenbanken	MySQL	Oracle
Browser	Firefox	MS Edge
Instantmessanger	Signal	WhatsApp
E-Mail Client	Thunderbird	MS Outlook
Sicherheitssoftware	VeraCrypt, GNUPG	PGP

4. Vor- und Nachteile von OSS gegenüber proprietärer Software

In diesem Abschnitt werden einige Vor- und Nachteile für und gegen den Einsatz von Open Source Software im Vergleich zur Proprietären Software aufgelistet.

4.1 Was spricht für die Nutzung von OSS?

Qualitätsnachweis: Durch die Zugänglichkeit des Quellcodes, lässt sich die Software auch vom Nutzer auf ihre Qualität untersuchen.

Man Power: Die Open Source Community hat eine hohe Anzahl an Mitgliedern die sich an der Entwicklung von Projekten beteiligen. Kein kommerziell orientiertes Unternehmen könnte eine vergleichbar große Zahl von Entwicklern bezahlen.

Anpassbarkeit: Nutzern ist es erlaubt Änderungen am Quellcode vorzunehmen, dadurch lässt sich das Programm an eigene Bedürfnisse anpassen.

Unabhängigkeit von einer Einzelperson oder Gruppe: Unterstützt ein Entwicklerteam die Weiterentwicklung eines Open Source Projekts, so kann ein anderes Team sich dazu entscheiden die Weiterentwicklung wieder aufzunehmen.

Geringe Kosten: Es fallen sehr wenig bis keine Lizenzgebühren an.

Wenig Schadsoftware: Da der Quelltext für alle einsehbar ist, lässt sich Schadsoftware schwieriger verstecken. Bei Betriebssystemen werden eher Betriebssysteme die eine größere Verbreitung haben angegriffen.

Ofene Standards: Offene Standards ermöglicht die nötige Interoperabilität zwischen den IT-Systemen verschiedener Hersteller. Der Kunde ist nicht durch ein sogenannten „Lock in Effekt“ an einem Hersteller gebunden.

4.2 Was spricht gegen die Nutzung von OSS?

Keine Garantie und professioneller Support: Dem Hersteller gegenüber besteht kein Anspruch auf Garantie oder Support Leistung.

Mängel bei Hardwareunterstützung: Die Hardware Unterstützung weist in manchen Fällen, z. B. bei Hardware-beschleunigten Grafikkarten oder bei Multimedia Equipment, wie Druckern, Mängel auf.

Keinen Verlass auf Weiterentwicklung: Eine Weiterentwicklung von Open Source Software gibt es keine Garantie, weil sie vom freiwilligen Engagement des Entwicklerteams sowie der Nachfrage abhängt.

4.3 Was spricht für die Nutzung von proprietärer Software?

Professioneller Support: Das Angebot an Support und Gewährleistung ist bei proprietärer Software professioneller, denn es können auch Vertragsstrafen für das Nichteinhalten von Vereinbarungen nach sich ziehen.

Planungssicherheit: Proprietäre Software bietet eine höhere Planungssicherheit, da die Weiterentwicklung der Software i. d. R. Vertraglich abgesichert ist.

Regelmäßige Updates und Patches: Hierbei wird zu festgelegten Zeitpunkten die Software aktualisiert zur Verbesserung sowie Erweiterung des Programms und mit „Patches“ werden Fehlerkorrekturen am Programm vorgenommen.

Geheimhaltung des Quellcodes: Dadurch, dass der Quellcode geheim gehalten wird, hat man eine erhöhte Sicherheit. So ist es schwieriger für Unbefugte den Quellcode zu manipulieren.

4.4 Was spricht die Nutzung proprietärer Software

Relativ hohe Kosten: Lizenzen proprietärer Software sind meistens Kostenpflichtig. Wartungsarbeiten und Support sind häufig schon in den Kosten mitenthalten.

Geheimhaltung des Quellcodes: Durch die Geheimhaltung des Quellcodes kann der Nutzer nicht ohne Weiteres die Software verändern bzw. an seine Bedürfnisse anpassen. Außerdem lässt sich der Quellcode nicht vom Nutzer auf ungewollte Lücken überprüfen.

Geschlossene Standards: Durch geschlossene Standards sind Softwareteile häufig inkompatibel zu Software anderer Hersteller. Dadurch sind Kunden an einem Hersteller gebunden („Lock in Effekt).

Abhängigkeit: Beschließt der Hersteller die Weiterentwicklung seiner Software einzustellen, muss der Kunde auf eine andere Software umsteigen.

5. Sicherheit

Im 5. Kapitel geht es um Die Sicherheit von OSS und proprietärer Software. Im ersten Abschnitt werden die Sicherheitskonzepte der zu vergleichenden Softwarearten kurz erläutert und im zweiten Abschnitt gegenübergestellt. Zusätzlich werden die Sicherheitsmaßnahmen Verschlüsselung von Daten und Anonymität im Netzt und deren Zusammenhang mit den beiden Software Arten kurz erklärt.

5.1 Sicherheitskonzept von OSS und proprietärer Software

Die Sicherheit von IT-Systemen wird immer wichtiger. Insbesondere durch die Sicherheitsskandale in den letzten Jahren, stieg das Bedürfnis für Sicherheit im Bereich Datenaustausch und -verwaltung. Beispielsweise wurde April 2014 eine Lücke in der quelloffenen Verschlüsselungsbibliothek Open SSL als „Heartbleed“ bekannt. Millionen von Webservern waren davon betroffen. Durch diese Sicherheitslücke konnten Häcker sensible Daten, wie etwa private Schlüssel und Passwörter aus Servern auslesen. Ähnlich dramatische Effekte hatte eine Sicherheitslücke im proprietären Code von Microsoft Windows, die im November 2014 bekannt wurde. Kriminelle nutzten sie, um die Kreditkarteninformationen von ca. 56 Millionen Konten des amerikanischen Einzelhändlers Home Depot zu stehlen. Microsoft schloss die Lücke erst, nachdem der Hack bekannt wurde.[23]
Sicherheitssysteme dienen dazu, Rechnersysteme und Netzwerke vor unbefugtem Zugriff zu schützen. Wenn es um IT-Sicherheit geht, dann werden i. d. R. drei wichtige Ziele verfolgt:

Integrität beim Kommunizieren. Hier ist gewünscht, dass die Nachrichten zwischen dem Sender und gewünschten Empfänger unverfälscht übermittelt werden. Abhilfe schafft hier z. B. das Erzeugen einer Prüfsumme über die verschickten Daten, die dann auf anderem Wege zum Empfänger gelangt. Durch Vergleich mit der aktuellen Prüfsumme beim Empfänger kann die Integrität festgestellt werden.

Vertraulichkeit zum Schutz der Privatsphäre. Die vertraulichen Informationen der Nutzer müssen vor dem Zugang durch andere geschützt sein. Da die Daten im Computernetzwerk leicht von anderen mitgelesen werden können, ist Vertraulichkeit nicht gewährleistet. Erst durch Verschlüsselung lässt sich Vertraulichkeit herstellen.

Verfügbarkeit von Systeme, Informationen und Funktionen darf nicht vorübergehend oder dauerhaft beeinträchtigt sein, sondern müssen verfügbar sein, wenn sie gebraucht werden. Um dies zu erreichen, können Datensicherung, Virenschutz und Verschlüsselung von Nutzen sein.

Sicherheitskonzept von OSS

Das zentrale Konzept von OSS ist die Offenlegung des Quellcodes. Dadurch kann jeder Nutzer das Programm auf dessen Qualität und mögliche Fehler untersuchen. Wird eine Lücke gefunden, so ist es dem Nutzer erlaubt, Änderungen und Verbesserungen an der Software vorzunehmen.
In der Praxis ist dies nur eingeschränkt möglich, da hierfür ein gewisses IT-Verständnis vonnöten ist, so dass i.d.R. nur versierte Nutzer und Experten dies auch tatsächlich tun. Doch auch einfache Nutzer einer OSS die nicht besonders IT-affin sind, können von der Offenlegung des Quellcodes profitieren. Denn in Internet gibt diverse File Hosting Online Dienste, Foren und Newsletter worüber OSS-Nutzer kommunizieren können. So lassen sich Fehler und Sicherheitslücken durch die Mitarbeit von sowohl Nichtprogrammierern als auch Programmierern in aller Welt schnell aufspüren und beheben.

[23] (Totzek-Hallhuber, 17)

Kein kommerziell orientiertes Unternehmen könnte eine vergleichbar große Zahl von Entwicklern bezahlen und so schnell reagieren. Durch das „Viele-Augen-Prinzip“, dass umso mehr Personen passiv und aktiv an der Software arbeiten, desto hochwertiger ist eine OSS, soll die Sicherheit gewährleistet werden.[24] So ist aufgrund des offenen Quellcodes, schwieriger „Trojaner“ zu verbergen.[25]
Um die Qualität von Änderungen an einer Software sicherzustellen, haben sich innerhalb Open Source Communtiy gewisse Verfahren bewährt. Es wird zunächst die Änderung mittels eines Patches eingereicht. Ein Patch beinhaltet dabei nur die geänderten Codes. Darauf wird die Änderung öffentlich diskutiert. Daraus folgt entweder die Annahme der Änderung und es werden Verbesserungen an ihr vorgenommen, oder sie wird abgelehnt. Anschließend wird nach Annahme der Änderung, die Änderung von einem Verantwortlichen des Projektes in den Quellcode eingepflegt und veröffentlicht.[26] Vor allem sicherheitsrelevante Tools wie Verschlüsselungsprogramme profitieren von diesem Konzept.
In so genannten Security Audits werden große Open Source Programme auf Probleme untersucht. ein Prozess, der bei proprietären Programmen mangels Quellcode schwer bis nicht zu realisieren ist.

Sicherheitskonzept von proprietärer Software

Bei proprietärer Software wird der Quellcode als Betriebsgeheimnis behandelt und somit von der Öffentlichkeit geheim gehalten. Hier wird angenommen, dass durch die Geheimhaltung des Quellcodes, Angreifern das Missbrauchen von Schwachstellen erschwert bzw. unmöglich wird. Diese Methode erschwert tatsächlich das Aufsuchen von Fehlern, unmöglich ist es jedoch nicht. Problematisch ist hierbei vor allem die nicht Überprüfbarkeit enthaltener „Backdoors“ für Geheimdienst oder weitere Unternehmen, da der Quelltext nicht einsehbar ist. Bei proprietärer Software erfolgt die Behebung von Fehlern von einer intern ausgewählten Gruppe von Entwicklern und wird den Nutzern häufig erst im Moment der Aktualisierung der Software mitgeteilt. Die Aktualisierung kann zu festgelegten Tagen erfolgen, sogenannte „Patch Days“.[27] Dies führt dazu, dass die bekannten Fehler und Schwachstellen für längere Zeit nicht behoben werden.

5.2 Sicherheitsvergleich

Beim Thema Sicherheit, ist das Hauptargument der Befürworter von Open Source Plattformen, dass im Gegensatz zur proprietären Software der Quelltext für alle zugänglich ist. Deshalb können Fehler durch Austausch schnell gefunden und korrigiert werden. Dieses Konzept setzt jedoch eine relativ große Gemeinschaft aus Nutzern und Entwicklern, die sich aktiv mit der OSS auseinandersetzen und untereinander Informationen austauschen, voraus. Zusätzlich braucht das Ganze eine gut überschaubare Struktur (z. B. Arbeitseinteilung, Dokumentation der Veränderung und etc.), damit das Ergebnis auch vor allem Stabil und Sicher ist.
Große bzw. bekannte Open Source Projekte erfüllen i. d. R. diese Kriterien und es lässt sich relativ leicht im Internet Hilfe bzw. Support finden. OSS die eher unbekannt und nur von einem sehr kleinen Entwicklerteam oder einer einzelnen Person erstellt wurden sind, haben ein höheres Risiko für Sicherheitsfehler und Schwächen in der Stabilität und ist schwieriger Hilfe zu finden, wenn sich nur wenige mit der Software beschäftigen. Lässt sich für eine bestimmte OSS keine zuverlässige Hilfe bei Rückfragen finden, gibt es die Alternative auf eine OSS umzusteigen, die eine große Nutzergemeinschaft hat oder man steigt auf eine proprietäre Alternative um. Letzteres bietet üblicherweise einen kostenpflichtigen Support bei Problemen und regelmäßige Updates um Fehler zu beheben.
Ein Vorteil an OSS ist die Unabhängigkeit der Software von einer einzelnen Organisation. Wenn die Weiterentwicklung einer OSS von einem Entwicklerteam aufgegeben wird, kann diese von einem anderen Team wieder aufgenommen werden. Beispielweise das Verschlüsselungstool TrueCrypt dass 2014 eingestellt wurde, konnte dessen Weiterentwicklung, durch die Zugänglichkeit des Quellcode,

[24] (Raymond, The Cathedral & the Bazaar, 2001)
[25] (Köhntopp, 2000)
[26] (Gehring, 2003)
[27] (Hofferbert, 2018)

von einem anderen Entwicklerteam weitergeführt werden. So entstand VeraCrypt.[28] Der einfache Nutzer ist auch hier wieder auf ein Entwicklerteam angewiesen, welches sich dafür bereiterklärt die Weiterentwicklung der OSS wiederaufzunehmen.
Dem gegenüber ist man Proprietärer Software darauf angewiesen, dass der Urheber die Weiterentwicklung der Software fortsetzt. Wird die Weiterentwicklung eingestellt, kann die Software nicht von jemand anderem übernommen werden. Generell bieten Proprietäre Software eine höhere Planungssicherheit als Open Source Software, denn die Weiterentwicklung im kommerziellen Rahmen kann vertraglich abgesichert werden (zumindest für einen gewissen Zeitraum). Sollten Entwickler keine Zeit oder kein Interesse mehr an ihrem Projekt besitzen, kann die Entwicklung eines Open Source Projekts jederzeit eingestellt werden.

5.3 Verschlüsselung & Anonymität

Mit Internet lassen sich dank modernster Netzwerktechnologien Informationen und Medien jeglicher Art in digitalisierter Form über digitale Nachrichtenkanäle transportieren und verbreiten. Gerade im Zeitalter der digitalen Kommunikation spielt die Verschlüsslung von Nachriten und das anonyme Surfen im Internet eine wichtige Rolle. Jede Seite die im Internet besucht wird, wird vermerkt. Auch wie lange der Aufenthalt ist und welche Seiten danach aufgerufen worden sind. Wenn z. B. Werbung angeklickt wird, wird dies von Werbeunternehmen gespeichert für, personenbezogene Werbung ausgewertet und eventuell weitergegeben. All diese Daten und weitere werden von Geheimdiensten auf Vorrat gehortet. Viele Menschen haben kein Problem damit. Andere, haben das Bedürfnis (und vor allem auch das Recht) sich unbeobachtet durch sowohl die digitale als auch reale Welt zu bewegen. Welche Möglichkeiten gibt es?

Verschlüsselung[29]

Unter Verschlüsselung versteht man Verfahren und Algorithmen, die Daten mittels elektronischer Codes inhaltlich in eine nicht lesbare Form umwandeln. Gleichzeitig wird dafür gesorgt, dass nur mit einem bestimmten Schlüssel die geheimen Daten wieder entschlüsselt werden können.
Beim Verschlüsseln wird der Klartext und ein Schlüssel an einem Verschlüsselungsalgorithmus (mathematische Funktion) übergeben. Mithilfe des Algorithmus entsteht ein Geheimtext (Chiffre), der keinen Rückschluss auf den Klartext erlaubt. Nur mit Kenntnis des Schlüssels kann man mit derselben mathematischen Funktion den Geheimtext wieder in den Klartext umwandeln.
Besonders bei Verschlüsselungssoftware zeigt sich das Open Source Konzept von Vorteil.
Durch die Offenlegung des Quellcodes, lässt sich die Funktionsweise des Verschlüsselungsalgorithmus einsehen. Und da das Verfahren bekannt ist, kann man es auf Fehler und Schwachstellen überprüfen. Auf diese Weise kann man sicherstellen, dass ein Verschlüsselungsalgorithmus für einen bestimmten Anwendungsfall sicher genug ist. Bei Proprietäre Software hingegen, kann man sich nicht sicher sein ob die Software keine Sicherheitslücken hat bzw. keine Geheime Türen für beispielsweise Geheimdienste hat, aus der Nutzerdaten abgefangen werden kann.
Für einen vollständigen Verschlüsselungsverfahren braucht es neben dem Algorithmus zum Verschlüsseln und Entschlüsseln ein Verfahren zum schlüsselaustausch und Prüfung der Authentizität und Integrität. Die bekannten Verfahren lassen sich in symmetrische und asymmetrische Verschlüsselungsverfahren einteilen. Bei der symmetrischen Verschlüsselung wird der Geheimtext mit demselben Schlüssel mit dem der Klartext verschlüsselt wurde wieder in den Klartext entschlüsselt. Nachteil beim symmetrischen Verfahren ist z. B. beim Verschicken von Nachrichten muss man sich um einen sicheren Schlüsselaustausch und deren Verteilung an den gewünschten Empfängern kümmern muss.
Die asymmetrische Kryptografie unterscheidet sich zur symmetrischen Kryptografie dadurch, dass die asymmetrische Kryptografie mit zwei Schlüsseln arbeitet. Einen zum Verschlüsseln und den anderen zum Entschlüsseln. Hierbei ist der Schlüssel zum Verschlüsseln öffentlich („Public Key“) und der Schlüssel zum Entschlüsseln bleibt geheim („Private Key“). Damit ein Sender einem Empfänger eine

[28] (Hofferbert, 2018)
[29] (Anonym, Verschlüsselung / Chiffrierung, kein Datum)

verschlüsselte Nachricht schicken kann, muss der Empfänger seinen öffentlichen Schlüssel dem Absender bekannt machen.
Der Schlüssel ist digital und ist eine Bitfolge, deren Länge in Bit angegeben ist. I. d. R. gilt, je länger ein Schlüssel ist, desto schwieriger ist es für Angreifer an das verschlüsselte Objekt zu gelangen. Sprich ein Schlüssel mit einer Länge von 1.024 Bit, also eine Folge von 1.024 Nullen und Einsen, ist sicherer als ein Schlüssel mit nur 64 Bit. Selbst wenn man weiß, wie verschlüsselt wird, müsste man alle möglichen Schlüssel durchprobieren, um irgendwann den richtigen Schlüssel zu bekommen. Eine absolute Sicherheit durch Verschlüsselung gibt es also nicht, lediglich der Aufwand wird erhöht. Mit Verschlüsselung gewinnt man also nur Zeit, bis jemand einen Weg findet den Geheimtext zu entschlüsseln. Eine starke Verschlüsselung ist laut Sicherheitsexperten sicher. Voraussetzung ist natürlich, dass die Schlüssel lang genug sind und dass der private geheime Schlüssel geheim ist und bleibt.

Anonymität im Netz[30]

Anonymität im Internet ist seit über zwei Dekaden ein großes Thema. Beim Surfen wird immer eine breite Datenspur hinterlassen. Durch Internetüberwachung kann auch ermittelt werden, wer über ein öffentliches Netzwerk mit wem kommuniziert.
Anonymitätssoftware hat die Aufgabe dies zu verhindern bzw. vor Internetüberwachung schützen. Um unbeobachtet durch das Netz zu Surfen gibt es mehrere Möglichkeiten. Eine davon ist Tor („The Onion Router“). Der Tor-Browser, welches eine auf Firefox basierende OSS ist, leitet jede Anfrage des Rechners verschlüsselt über eine Kette von Tor-Rechnern weltweit, dem sogenannten Onion-Netzwerk, um. Das Besondere hierbei ist, das der jeweilige Tor-Rechner nicht in die verschlüsselten Daten der anderen Tor-Rechner blicken kann, sprich keiner der verbundenen Tor-Rechner weis wer der Nutzer ist und wohin die Anfrage geht. Dadurch kann der Webserver nur die jeweils letzte IP-Adresse in der Kette sehen und für Angreifer ist es so gut wie unmöglich die Adresse zurückzuverfolgen.

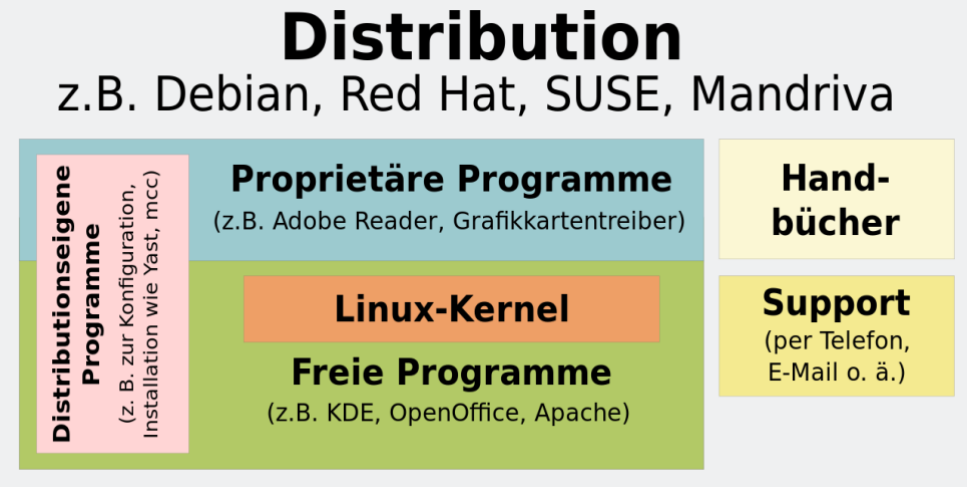
[30] (Anon, kein Datum)

6. Geschäftsmodelle der OSS

Das Ziel von Richard Stallmann, ein Software Universum wiederherzustellen, in der Software quelloffen vorliegt und jeder Software frei und uneingeschränkt verwenden kann, wurde mit sehr viel Kritik begegnet. Vor allem Hersteller proprietärer Software kritisierten dieses Vorhaben mit dem Argument, dass Software Entwickler kein Geld an freier Software verdienen würden. Darauf antwortete Richard Stallmann in seinem „GNU-Manifest“, man könne Geschäftsmodelle um quelloffene Software bilden an der auch Entwickler verdienen können. Heute gibt es sehr viele Unternehmen die dies erfolgreich umgesetzt haben. In diesem Kapitel werden einige dieser Modelle kurz veranschaulicht. Viele dieser Geschäftsmodelle kommen nicht in Einzelform vor, sondern sind gemischte Modelle (So können z. B. viele Distributoren gleichzeitig auch Dienstleister sein).

6.1 Distributoren

Distributoren erstellen Software Pakete auf Datenträgern, die Programme für die unterschiedlichsten Einsatzgebiete enthalten und machen sie durch Installations- und Administrationsroutinen für Endkunden als Komplettprodukt nutzbar. Diese Art von Zusammenstellung von Softwarekomponenten wird Distribution genannt. Das Geschäftsmodell der Distributoren besteht aus Entwicklung, Vermarktung, Vertrieb und Support dieser Distributionen. Zu den in Deutschland bekanntesten Distributoren zählen SuSE Linux und Red Hat. Viele Distributoren bieten auch Support und Beratung an.



Quelle(Gissi, Phrood, & Streb, kein Datum)

6.2 Appliance Entwickler

Appliances sind funktionelle Einheiten, die aus einer Hardware- und Software-Kombination bestehen. Diese werden im Speziellen für eine konkrete Aufgabe konzipiert, wodurch sie eine hohe Sicherheit und Stabilität erhalten.
Die Anbieter entwickeln häufig zusätzliche eigene Applikationen die für Hardware und auch für die Schnittstelle zum Benutzer (Bedienoberfläche, Aktualisierungen und ähnliches) speziell optimiert sind. Diese Applikationen können proprietär sein, was aber nicht gegen der GPL widerspricht, da die Software zwar zusammen aufgespielt ist, aber nicht technisch aufeinander aufbaut.

Beispiele für Appliance Anbieter:

- Server-Appliances: IBM Whistle, eSoft
- Thin-Clients: IBM NetVista,
- Set-top-boxen: Nokia, PersonalTV,

6.3 Dienstleistungsbereich

Lange Zeit wurde mangelnder Support als Hauptargument gegen den Einsatz von OSS gesehen. mittlerweile hat sich dies deutlich gebessert. Sehr viele Unternehmen bieten heute Dienstleistungen um OSS an. Die Beratung rund um das Thema Open Source Programme zählt zu den wesentlichen Dienstleistungen dieser Firmen. Sie analysieren für ihre Kunden, welche Produkte für die gewünschten Bereiche sinnvoll sind und installieren anschließend die passende Software in das bestehende System-Umfeld. Zusätzlich bieten sie professionellen Support an und sind für weitere kundenspezifische Anpassungen zuständig. Einige bieten Schulungen an um den Kunden in die Software einzuführen

7. Fazit

In der Vergangenheit stieß quelloffene Software auf viele Gegner, darunter hauptsächlich kommerzielle Software Unternehmen die das Konzept stark kritisierten und auch ablehnten.
Nun hat Open Source die Softwareindustrie umgekrempelt wie kaum eine Idee zuvor. Mittlerweile ist Open Source nicht mehr aus der Softwareindustrie wegzudenken. Heute gibt's es in fast jeder Software Kategorie eine Open Source Alternative zur proprietären Software. Nahezu jeder ist schon mal auf OSS gestoßen. Open Source wird oft als Angriff gegen die kommerzielle Welt dargestellt. Jedoch zeigt sich, dass OSS ein Segen für alle ist, denn jeder profitiert davon. Sei es ein Student, der eine kostengünstige stabile Software benötigt, um eine Seminararbeit zu fertigen oder ein Hobby Entwickler, der Anpassungen an einer Software vornehmen möchte. Auch kommerzielle Unternehmen profitieren davon, denn die gemeinschaftliche Entwicklung von Projekten spart Ressourcen und erlaubt flexible Weiterentwicklungen der Technologie. Es lässt sich sogar sagen, dass sowohl proprietäre als auch quelloffene Software miteinander harmonieren. Beispielsweise das weltweit am meisten verbreitete Smartphone Betriebssystem Android von Google, stammt vom Linux-Kernel ab und dient als Plattform für viele Applikationen, die proprietär vertrieben werden. Viele Open Source Projekte werden von kommerziellen Unternehmen finanziert und auch geleitet. OSS kann in fast allen Kriterien mit proprietärer Software mithalten. Für einige Bereiche eignet sich OSS und für andere proprietäre Software mehr. Es gibt keinen klaren Gewinner zur Frage, welche Softwareart nun besser sei. Beide Arten haben ihre Vor- und Nachteile. Es kommt also auf die Situation an. Dies gilt auch für das Thema Stabilität und Sicherheit. Es reicht nicht einfach nur aus, dass der Quelltext bei OSS für alle zugänglich oder bei proprietärer Software unzugänglich ist. Wie sicher und zuverlässig ein Programm arbeitet, hängt immer auch davon ab, wer für die Entwicklung zuständig ist, das Programm nutzt und für was es verwendet wird. Ein sicherheitsrelevantes Programm, an dem seit Jahren nicht mehr gearbeitet wird, sollte auch bei offenem Quellcode mit Skepsis betrachtet werden. Ist der Code hingegen sauber dokumentiert und befassen sich viele Programmierer mit der Weiterentwicklung ist die Chance, ein sicheres Open Source Projekt vor sich zu haben, relativ hoch. Wahrscheinlich sogar sicherer und stabiler als proprietäre Alternativen. Bezüglich der Sicherheit von Betriebssystems wird Beispielsweise der Vergleich Linux zu Windows aufgeführt. Über die Jahre gab es kaum Schadsoftware für Linux. Das heißt nicht unbedingt, dass Linux deutlich sicherer ist als Windows. Dies könnte man darauf zurückführen, dass Linux einen geringeren Marktanteil hat, was Linux-Nutzer zu einem unattraktiveren Ziel macht. Auch bei einer OSS die von Millionen von Menschen genutzt wird, können immer noch Sicherheitslücken vorliegen („Heartbleed Bug"). Die volle Sicherheit einer Software ist nie gegeben, denn diese ist nur so lange sicher, bis eine Sicherheitslücke erkannt und missbraucht wird.
Personen die darauf angewiesen sind oder einfach sehr viel Wert darauflegen, ihre Identität oder Nachrichten und Daten geheim zu halten, sind OSS eher zu empfehlen. In den letzten Jahren hat sich gezeigt, dass kommerzielle Unternehmen nicht sehr vertrauenswürdig mit Nutzerdaten umgehen („PRISM-Skandal"). Durch die Offenlegung des Quellcodes kann die OSS auf „Backdoors" für beispielsweise Geheimdienste überprüft werden.

Literaturverzeichnis

Anon. (kein Datum). *Tor: Overview*. Abgerufen am 2018 April von https://www.torproject.org/about/overview.html.en

Anonym , A. (besucht 2018). *Open Source Defnition.* http://deacademic.com/dic.nsf/dewiki/1052582#sel=.

Anonym. (Mai 2018). *Licenses & Standards*. Von https://opensource.org/licenses abgerufen

Anonym. (kein Datum). *Ofene Standards*. Abgerufen am April 2018 von https://fsfe.org/activities/os/os.de.html

Anonym. (kein Datum). *The Apache Software Foundation*. Abgerufen am April 2018 von http://apache.org/foundation/how-it-works.html

Anonym. (kein Datum). *The Linux Kernel Archives*. Abgerufen am Mai 2018 von https://www.kernel.org/category/about.html

Anonym. (kein Datum). *Verschlüsselung / Chifrierung*. Abgerufen am April 2018 von https://www.elektronik-kompendium.de/sites/net/1907041.htm

Breeding, M. (2002). *The open source ILS: still only a distant possibility?* Chicago, IL: American Library Association.

Gates, B. (1976). *Open Letter to Hobbyists.*

Gehring, R. (20. März 2003). *Open Source Software - Sicherheit im Spannungsfeld von Ökonomie und Politik.* Abgerufen am April 2018 von http://einst-ig.de/sites/einst-ig.de/files/Gehring-OpenSourceSicherheit-032003.pdf

Gissi, Phrood, & Streb, E. (kein Datum). *Bestandteile einer Linux-Distribution*. Abgerufen am April 2018 von https://de.wikipedia.org/wiki/Linux#/media/File:Linux-Distribution.svg

Grassmuck, V. (2004). *Freie Software Zwischen Privat- und Gemeineigentum*. Bonn: Bundeszentrale für politische Bildung.

Hoch, D. J., Roeding , C., & Purkert, G. (1999). *Secrets of Software Success: Management Insights from 100 Software Firms Around the World*. Harvard Business Review Press.

Hofferbert, B. (2. Januar 2018). *Ist Open Source Software wirklich sicherer?* Abgerufen am April 2018 von https://www.heise.de/tipps-tricks/Ist-Open-Source-Software-wirklich-sicherer-3929357.html

Jaeger, T., & Metzger , A. (2002). *Open Source Software: Rechtliche Rahmenbedingungen der Freien Software*. München: C.H.Beck.

Köhntopp, K. (2000). Sicherheit durch Open Source? Chancen und Grenzen. In *Datenschutz und Datensicherheit (DuD)*. Wiesbaden: Vieweg.

Lang, M., & Augsten, S. (Mai 2017). *Software korrekt lizenzieren: Open-Source-Lizenzen rechtskonform einsetzen*. Abgerufen am Mai 2018 von https://www.dev-insider.de/open-source-lizenzen-rechtskonform-einsetzen-a-604458/

Lerner, J., & Tirole, J. (2000).

Levy, S. (2001). *Hackers: Heroes of the Computer Revolution.* Penguin.

Martin-Jung, H., & Neidhart, C. (2018). *Mit Open Source siegt die Vernunft*. Süddeutsche Zeitung.

Moody, G. (2002). *Rebel Code: Linux And The Open Source Revolution*. Basic Books.

Mundhenke, J. (2007). *Wettbewerbswirkung von Open-Source-Software und ofenen standards auf Softwaremärkten*. Berlin: Sringer.

Perence , B. (1999). *The Open Source Definition*. Abgerufen am April 2018 von https://opensource.org/osd

Raymond, E. (1999). *A Brief History of Hackerdom*. Abgerufen am April 2018 von http://www.catb.org/esr/writings/homesteading/hacker-history/index.html

Raymond, E. (1999). *The Revenge of the Hackers, Voices from the Open Source Revolution* (1. Auflage). O'Reilly & Associates.

Raymond, E. (2001). *The Cathedral & the Bazaar*. O'Reilly and Associates.

Raymond, E. (2001). *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly and Associates.

Stallman, R. (2002). *Why Open Source misses the point of Free Software.* https://www.gnu.org/philosophy/open-source-misses-the-point.en.html.

Torvalds , L., & Diamond , D. (2001). *Just for Fun - Wie ein Freak die Computerwelt revolutionierte.* Hanser Fachbuch.

Totzek-Hallhuber, J. (30. 01 17). *Open Source vs. Closed Source: Was ist sicherer?* Abgerufen am 17. April 2018 von https://www.security-insider.de/open-source-vs-closed-source-was-ist-sicherer-a-576230/

Williams, S. (2002). *Free as in Freedom: Richard Stallman's Crusade for Free Software.* O'Reilly Media.