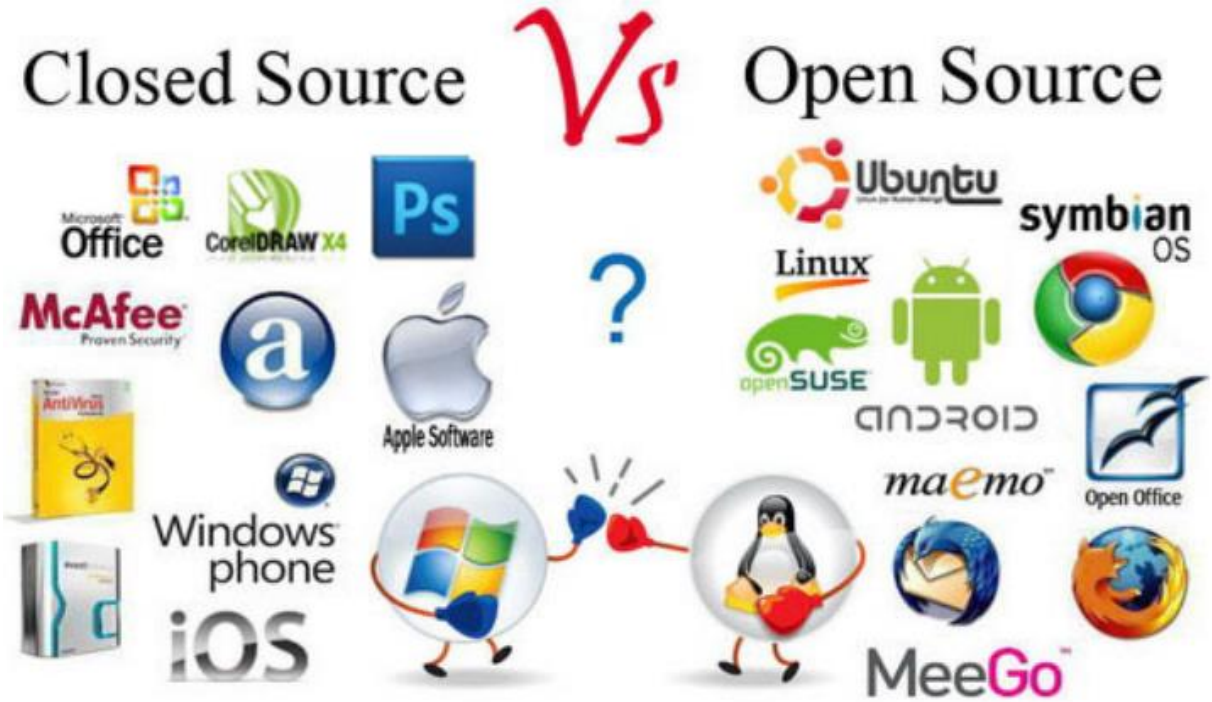
Thema 1: Die Linux-Community und Karriere im Open-Source-Umfeld

# 1.3 Lektion 1

**Zertifikat:** Linux Essentials
**Version:** 1.6
**Thema:** 1 Die Linux-Community und Karriere im Open-Source-Umfeld
**Lernziel:** 1.3 Open-Source-Software und -Lizenzen
**Lektion:** 1 von 1

## Einführung

Während die Begriffe ***freie Software*** und ***Open Source Software*** weit verbreitet sind, gibt es immer noch einige Missverständnisse über ihre Bedeutung. Insbesondere das Konzept der “Freiheit” bedarf einer näheren Betrachtung. Beginnen wir mit der Definition der beiden Begriffe.



## Definition von freier und Open Source Software

Kriterien für freie Software

Zunächst einmal hat “frei” im Kontext freier Software nichts mit “kostenlos” zu tun, oder wie der Gründer der ***Free Software Foundation* (FSF),** Richard Stallman, es prägnant ausdrückt:

Um das Konzept zu verstehen, sollten Sie bei “frei” an “Redefreiheit” denken, nicht an “Freibier”.

— Richard Stallman: *Was ist freie Software?*

Unabhängig davon, ob Sie für die Software bezahlen müssen oder nicht, gibt es **vier Kriterien, die freie Software ausmachen.** Richard Stallman beschreibt diese Kriterien als “die vier wesentlichen Freiheiten”, deren Zählung bei null beginnt:

- **“Die Freiheit, das Programm auszuführen wie man möchte, für jeden Zweck (Freiheit 0).”**

  Wo, wie und zu welchem Zweck die Software eingesetzt wird, kann weder vorgeschrieben noch eingeschränkt werden.

- “Die Freiheit, die **Funktionsweise des Programms** zu untersuchen und eigenen Datenverarbeitung-Bedürfnissen **anzupassen (Freiheit 1).** Der **Zugang zum Quellcode** ist dafür Voraussetzung.”

  Jeder kann die Software nach seinen Vorstellungen und Bedürfnissen ändern. Dies wiederum setzt voraus, dass der sogenannte *Quellcode*, d.h. alle Dateien, aus denen eine Software besteht, müssen in einer für Programmierer lesbaren Form vorliegen. Und natürlich gilt dieses Recht für einen einzelnen Benutzer, der eine einzelne Funktion hinzufügen möchte, sowie für Software-Unternehmen, die komplexe Systeme wie Smartphone-Betriebssysteme oder Router-Firmware erstellen.

- **“Die Freiheit, das Programm zu redistribuieren und damit Mitmenschen zu helfen (Freiheit 2).** Der Zugang zum Quellcode ist dafür Voraussetzung.”

  Diese Freiheit ermutigt jeden Benutzer ausdrücklich, die **Software mit anderen zu teilen.** Es geht also um die größtmögliche Verbreitung und damit die größtmögliche Gemeinschaft von Nutzern und Entwicklern, die auf der Grundlage dieser Freiheiten die Software zum Wohle aller weiterentwickeln und verbessern.

- **“Die Freiheit, das Programm zu verbessern und diese Verbesserungen der Öffentlichkeit freizugeben,** damit die gesamte Gesellschaft davon profitiert **(Freiheit 3).** Der Zugang zum Quellcode ist dafür Voraussetzung.”

  Es geht also nicht nur um die Verbreitung von freier Software, sondern auch um die **Verbreitung von *veränderter* freier Software**. Jeder, der Änderungen an freier Software vornimmt, hat das Recht, die Änderungen anderen zugänglich zu machen. Wenn man dies tut, ist man auch dazu verpflichtet, d.h. man darf die ursprünglichen Freiheiten bei der Verbreitung der Software nicht einschränken, auch wenn man sie verändert oder erweitert hat. Wenn eine

Gruppe von Entwicklern beispielsweise unterschiedliche Vorstellungen von der künftigen Ausrichtung einer Software als die ursprünglichen Autoren hat, kann sie ihren **eigenen Entwicklungszweig (*Fork* genannt) abspalten** und als neues Projekt fortsetzen. Aber natürlich bleiben alle Verpflichtungen im Zusammen-hang mit diesen Freiheiten bestehen.

Die Betonung des Freiheitsgedankens ist auch insofern konsequent als sich jede Freiheitsbewegung *gegen* etwas richtet, nämlich einen Gegner, der die postulierten Freiheiten unterdrückt, der Software als Eigentum betrachtet und unter Verschluss halten will. Im **Gegensatz zu freier Software** wird solche Software als ***proprietär*** bezeichnet.

## Open Source Software vs. freie Software

Für viele sind *freie Software* und *Open Source Software* Synonyme. Die häufig verwendete Abkürzung ***FOSS* für *Free and Open Source Software*** betont genau diese Gemeinsamkeit. **FLOSS für *Free/Libre and Open Source Software*** ist eine weitere beliebte Begriffsvariante, die den Freiheitsgedanken auch für andere Sprachräume als den englischen unmissverständlich herausstellt. Betrachtet man allerdings Entstehung und Entwicklung beider Begriffe, lohnt sich eine Differenzierung.


Der Terminus ***freie Software*** mit der Definition der beschriebenen vier Freiheiten geht auf **Richard Stallman** und das bereits 1985 — also beinahe 10 Jahre vor der Entstehung von Linux — von ihm gegründete Projekt *GNU* zurück. Der Name “**GNU is not Unix**” beschreibt augenzwinkernd die Intention: GNU startete als Initiative, eine technisch überzeugende, aber bis dahin allein der Kontrolle von Konzernen unterliegende Lösung — nämlich das **Betriebssystem Unix — von Grund auf neu zu entwickeln, der Allgemeinheit zur Verfügung zu stellen** und mit der Allgemeinheit

laufend zu verbessern. Die Offenheit des Source Code war dafür lediglich eine technisch-organisatorische Notwendigkeit — in ihrem Selbstverständnis ist die Free-Software-Bewegung aber bis heute eine *soziale* und *politische* — manche sagen auch ideologische — Bewegung.

Mit dem Erfolg von Linux, den kollaborativen Möglichkeiten des Internet und den Tausenden Projekten und Firmen, die in diesem neuen Software-Kosmos entstanden, trat der soziale Aspekt immer öfter in den Hintergrund. Die Offenheit des Quellcodes wurde von einer technischen Voraussetzung selbst zu einem Definitionsmerkmal: **Sobald der Quellcode einsehbar war, galt die Software als Open Source.** Die sozialen Motive wichen einem eher pragmatischen Ansatz der Softwareentwicklung.

Dieses Spannungsverhältnis besteht bis heute: **Freie Software** und **Open Source** Software **arbeiten an derselben Sache**, mit denselben Methoden und in einer weltweiten Community aus Einzelpersonen, Projekten und Firmen. Aber da sie soz. aus **verschiedenen Richtungen** — nämlich **einer sozialen und einer pragmatisch-technischen** — zusammengefunden haben, kommt es bisweilen zu Konflikten. Diese Konflikte entstehen, wenn die Ergebnisse der gemeinsamen Arbeit nicht den ursprünglichen Zielen *beider* Bewegungen entsprechen. Das geschieht vor allem dann, wenn eine **Software zwar ihre Quellen offenlegt, aber nicht zugleich die vier Freiheiten der freien Software respektiert**, wenn es also beispielsweise bei der Offenlegung, bei der Veränderung oder bei den Verbindungen mit anderen Software-Komponenten Beschränkungen gibt.

**Welchen Bedingungen eine Software hinsichtlich Nutzung, Weitergabe und Veränderung unterliegt, bestimmt die *Lizenz*, unter der die Software steht.** Und weil Anforderungen und Motive sehr unterschiedlich sein können, sind im FOSS-Bereich zahllose **verschiedene Lizenzen entstanden**. Aufgrund des deutlich fundamentaleren Ansatzes der Free-Software-Bewegung nimmt es nicht Wunder, dass sie viele Open-Source-Lizenzen nicht als “frei” anerkennt und darum abgelehnt. Umgekehrt ist das aufgrund des deutlich pragmatischeren Open-Source-Ansatzes naturgemäß kaum der Fall. Werfen wir im Folgenden einen kurzen Blick auf das tatsächlich sehr komplexe Gebiet der Lizenzen.

## Lizenzen

Anders als ein Kühlschrank oder ein Auto ist **Software** kein physisches, sondern ein **digitales Produkt.** Eine Firma durch Verkauf ein solches Produkt also nicht physisch übergeben — sie **übergibt vielmehr die Nutzungsrechte** an diesem Produkt, und der Nutzer stimmt vertraglich diesen Nutzungsrechten zu. Welche Nutzungsrechte das sind und vor allem *nicht* sind, ist in der **Software-Lizenz festgehalten**, und damit wird verständlich, welche Bedeutung den darin festgehaltenen Regelungen zukommt.

Während große Anbieter **proprietärer Software wie Microsoft oder SAP** eigene, genau **auf ihre Produkte abgestimmte Lizenzen haben**, waren die Verfechter freier und quelloffener Software von Anfang an um Klarheit und Allgemeingültigkeit ihrer Lizenzen bemüht, denn schließlich soll jeder Benutzer diese verstehen und gegebenenfalls selbst für seine eigenen Entwicklungen nutzen.

Es sei allerdings nicht verschwiegen, dass dieses Ideal der Einfachheit kaum zu erreichen ist, denn zu viele spezifische Anforderungen einerseits und international nicht immer kompatible Rechtsverständnisse stehen dem entgegen. Um nur ein

Beispiel zu nennen: **Deutsches und amerikanisches Recht unterscheiden** sich fundamental im Urheberrecht. Nach **deutscher Rechtsauffassung** gibt es **eine *Person* als *Urheber*, deren Werk ihr *geistiges Eigentum*** ist. Der Urheber kann zwar die Erlaubnis zur Nutzung seines Werkes erteilen, kann aber seine **Urheberschaft nicht abtreten oder aufgeben**.

Bei **amerikanischen Recht** gibt es zwar auch einen *Autor* (der allerdings auch eine Firma oder eine Institution sein kann), aber er verfügt lediglich über **Verwertungsrechte**, die er in **Teilen oder vollständig übertragen und sich damit vollständig von seinem Werk lösen kann**. Eine international gültige Lizenz muss vor solch unterschiedlichen Rechtsauffassungen interpretierbar sein.

Die Folge sind zahlreiche und zum Teil sehr **verschiedene FOSS-Lizenzen**. Und was noch schlimmer ist: Verschiedene Versionen einer Lizenz oder die **Mischung verschiedener Lizenzen** (innerhalb eines Projekts oder auch bei der Verbindung mehrere Projekte) sorgen bisweilen für Verwirrung oder gar juristische Streitfälle.

## Copyleft

Die bereits erwähnte ***Free Software Foundation*** **(FSF)** hat mit der ***GNU General Public License*** **(GPL) eine der wichtigsten Lizenzen für freie Software formuliert**, die viele Projekte, z.B. auch der Linux-Kernel, nutzen.

Darüber hinaus hat sie weitere Lizenzen mit **fallspezifischen Anpassungen** veröffentlicht, etwa die ***GNU Lesser General Public License*** **(LGPL)**, die die bisweilen unvermeidliche Kombination freier Software mit weniger freien Komponenten regelt, die ***GNU Affero General Public License*** **(AGPL)**, die den Verkauf des Zugangs zu gehosteter Software regelt, oder die ***GNU Free Documentation License*** **(FDL)**, mit der sie die freiheitlichen Grundsätze auf die Dokumentation von Software ausdehnt. Darüber hinaus spricht die FSF Empfehlungen für oder gegen Lizenzen Dritter aus, und angegliederte Projekte wie GPL-Violations.org gehen beispielsweise Verdachtsfällen nach, in denen es um die Verletzung freier Lizenzen geht.

Das **Prinzip**, nach dem sich eine **freie Lizenz** auch auf veränderte Varianten der **Software überträgt**, bezeichnet die **FSF als *Copyleft*** — im **Gegensatz** zu dem von ihr abgelehnten Prinzip des **restriktiven Copyrights**.

Die Idee ist also, die freiheitlichen Grundsätze einer Software-Lizenz möglichst uneingeschränkt auf künftige Bearbeitungen der Software zu übertragen, um nachträgliche Restriktionen zu verhindern. Was naheliegend und einfach klingt, führt in der **Praxis** jedoch zu zum Teil **erheblichen Komplikationen**, weshalb das Copyleft-Prinzip von Kritikern häufig auch als “viral” bezeichnet wird, da es sich auf Folgeversionen überträgt.

Aus dem Gesagten folgt zum Beispiel, dass sich zwei Software-Komponenten, die unter **verschiedenen Copyleft-Lizenzen stehen, nicht miteinander kombinieren lassen**, da sich ja nicht beide Lizenzen gleichzeitig auf das Folgeprodukt übertragen lassen. Das kann sogar für verschiedene Versionen derselben Lizenz gelten!

Aus diesem Grunde fassen neuere Lizenzen oder Lizenz-Versionen das Copyleft oftmals nicht mehr so rigoros. Schon die genannte ***GNU Lesser General Public License* (LGPL)** ist in diesem Sinne ein Zugeständnis, um **freie Software überhaupt mit "unfreien" Komponenten verbinde**n zu können, wie es zum Beispiel häufig bei *(Programm-)Bibliotheken* (engl. *libraries*) geschieht.

Eine weitere Möglichkeit zur Vermeidung von Lizenzkonflikten ist das ***Dual Licensing*,** bei dem eine **Software unter verschiedenen Lizenzen steht,** z.B. einer freien und einer proprietären. Ein typischer Anwendungsfall ist eine kostenlose Version einer Software, die nur unter Beachtung der Copyleft-Beschränkungen genutzt werden darf, und des alternativen Angebots, die Software unter einer anderen Lizenz zu beziehen, was den Lizenznehmer von bestimmten Einschränkungen befreit, und zwar gegen eine Gebühr, die zur Finanzierung der Entwicklung der Software verwendet werden könnte.

## Open Source Definition und permissive Lizenzen

Auf Seiten der **Open-Source-Bewegung** ist es die **1998** von **Eric S. Raymond** und **Bruce Perens** gegründete ***Open Source Inititive* (OSI)**, die sich maßgeblich mit Fragen der Lizenzierung beschäftigt. Sie hat auch ein standardisiertes Verfahren entwickelt, mit dem sie Software-Lizenzen auf ihre Übereinstimmung mit der von ihr formulierten *Open Source Definition* überprüft. Auf der Website der OSI (Link: https://opensource.org/ ) finden sich aktuell **mehr als 80 anerkannte Open-Source-Lizenzen.**

Auszug aus Open Source Lizenzen (OSI-Website)

- Apache License 2.0
- BSD 3-Clause "New" or "Revised" license
- BSD 2-Clause "Simplified" or "FreeBSD" license
- GNU General Public License (GPL)
- GNU Library or "Lesser" General Public License (LGPL)
- MIT license
- Mozilla Public License 2.0
- Common Development and Distribution License
- Eclipse Public License version 2.0

Hier gibt es auch **Lizenzen**, die dem **Copyleft-Prinzip ausdrücklich widersprechen**, allen voran die Gruppe der ***BSD-Lizenzen***. Die *Berkeley Software Distribution* (BSD) bezeichnet eine ursprünglich an der Universität Berkeley entwickelte Variante des Betriessystems Unix, aus denen später wiederum freie Projekte wie *NetBSD*, *FreeBSD* und *OpenBSD* hervorgingen.

Die diesen Projekten zugrundeliegenden Lizenzen bezeichnet man häufig als *permissive* ("freizügig"). Im Gegensatz zu Copyleft-Lizenzen verfolgen sie gerade *nicht* das Ziel, auch die Nutzungsbedingungen veränderter Varianten festzuschreiben. Das Höchstmaß an Freizügigkeit soll der Software vielmehr zu einer möglichst weiten Verbreitung verhelfen, indem allein den Bearbeitern der Software überlassen wird, wie sie mit den Bearbeitungen verfahren — ob sie sie beispielweise ebenfalls freigeben oder auch als Closed Source behandeln und kommerziell vertreiben.

## Creative Commons

Das erfolgreiche **Entwicklungskonzept von FLOSS** und die damit verbundenen technologischen Fortschritte sorgten dafür, dass man das Open-Source-Prinzip auch auf andere, **nicht-technische Bereiche zu übertragen** versuchte. Die Aufbereitung und Bereitstellung von Wissen wie auch das kreative Miteinander bei der Lösung komplexer Aufgaben gelten heute als Belege für das erweiterte, inhaltsbezogene Open-Source-Prinzip.

Damit entstand die Notwendigkeit, auch in diesen Bereichen verlässliche Grundlagen zu schaffen, nach denen Arbeitsergebnisse geteilt und bearbeitet werden können. Da die vorliegenden Software-Lizenzen dafür kaum geeignet waren, gab es zahlreiche Versuche, die die spezifischen Anforderungen von **wissenschaftlichen Arbeiten bis hin zu digitalisierten Kunstwerken "im Geist von Open Source" in ähnlich griffige Lizenzen überführten.**

Die heute mit Abstand wichtigste Initiative dieser Art ist ***Creative Commons* (CC)**, die ihr Anliegen selbst wie folgt zusammenfasst:

**Creative Commons ist eine weltweite Non-Profit-Organisation, die das Teilen und Wiederverwenden von Kreativität und Wissen durch die Bereitstellung freier juristischer Hilfsmittel ermöglicht**.

— https://creativecommons.org/faq/#what-is-creative-commons-and-what-do-you-do

Mit Creative Commons geht der Fokus der **Rechtevergabe** vom Distributor zurück **auf den Urheber/Autor**. Ein Beispiel: Im klassischen Verlagswesen überträgt ein Urheber meist sämtliche Verwertungsrechte (Druck, Übersetzung etc.) an einen Verlag, der im Gegenzug für die bestmögliche Verbreitung des Werks sorgt. Die deutlich veränderten Distributionswege des Internets versetzen nun den Urheber in die Lage, viele dieser Verwertungsrechte selbst auszuüben und selbst zu entscheiden, wie sein Werk genutzt werden darf.

**Creative Commons** gibt ihm die Möglichkeit, dies einfach und juristisch verlässlich festzulegen, will aber mehr: Der Urheber wird ermutigt, sein Werk als Beitrag einem allgemeinen Prozess des Austauschs und der Zusammenarbeit zur Verfügung zu stellen. Anders als das traditionelle Copyright, das dem Urheber sämtliche Rechte sichert, die er nach Bedarf an andere überträgt, wählt der Creative-Commons-Ansatz den umgekehrten Weg: **Der Urheber stellt sein Werk der Gemeinschaft zur Verfügung, kann aber aus einem Satz von Eigenschaften jene auswählen, die bei der Nutzung des Werks zu beachten sind** — je mehr Eigenschaften er auswählt, desto restriktiver die Lizenz.

Zum besseren Verständnis, hier ein Überblick über die sechs von CC angebotenen Kombinationsmöglichkeiten bzw. Lizenzen:

- **CC BY (“Namensnennung”)**

  Die **freieste Lizenz**, nach der jeder das Werk bearbeiten und weitergeben darf, **solange er den Urheber nennt**.

- **CC BY-SA (“Namensnennung - Weitergabe unter gleichen Bedingungen**”)

Wie BY, nur dass das veränderte Werk ausschließlich **unter derselben Lizenz weitergegeben werden darf**. Das Prinzip **erinnert an das Copyleft**, da auch hier die Lizenz "vererbt" wird.

- **CC BY-ND ("Namensnennung-Keine Bearbeitung")**

  Wie CC BY, nur dass das Werk ausschließlich **unverändert/unbearbeitet weitergegeben** werden darf.

- **CC BY-NC ("Namensnennung-Nicht kommerziell")**

  Das Werk darf unter Nennung des Urhebers zwar **bearbeitet und weitergegeben werden**, allerdings ausschließlich unter **nicht-kommerziellen Bedingungen.**

- **CC BY-NC-SA ("Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen")**

  Wie BY-NC, nur dass das Werk ausschließlich **unter denselben Bedingungen weitergegeben werden darf** (auch hier also eine Copyleft-ähnliche Lizenz).

- **CC BY-NC-ND ("Namensnennung - Nicht-kommerziell - Keine Bearbeitung")**

  Die **restriktivste Lizenz**: die Weitergabe ist mit Namensnennung des Urhebers erlaubt, allerdings **nur unverändert und unter nicht-kommerziellen** Bedingungen.

## Business-Modelle in Open Source

Im FLOSS-Umfeld sind milliardenschwere Unternehmen entstanden; stellvertretend sei hier die 1993 gegründete US-amerikanische **Firma *Red Hat*** mit einem **Jahresumsatz von über 3 Milliarden USD (2018)** genannt, die 2018 vom IT-Giganten IBM übernommen wurde.

Werfen wir also einen Blick auf das Spannungsverhältnis zwischen der freien und tatsächlich unentgeltlichen Weitergabe hochwertiger Software und den Business-Modellen für ihre Schöpfer, denn eines sollte klar sein: Die zahllosen hochqualifizierten Entwickler freier Software müssen auch Geld verdienen, und das ursprünglich tatsächlich rein nicht-kommerzielle FLOSS-Umfeld muss folglich nachhaltige Business-Modelle entwickeln, um seinen eigenen Kosmos zu bewahren.

Ein häufiger Ansatz, vor allem für größere Projekte in der Anfangsphase, ist das so genannte ***Crowdfunding***, also das Einsammeln von Geldspenden über eine Plattform wie *Kickstarter*. Die Spender erhalten dafür von den Entwicklern im Erfolgsfalle, also bei Erreichen zuvor definierter Ziele, eine zuvor festgelegte Gratifikation, seien dies ein unbeschränkter Zugang zum Produkt oder spezielle Features.

Ein anderer Ansatz ist das ***Dual Licensing***. Eine freie Software wird parallel unter einer restriktiveren oder gar proprietären Lizenz angeboten, die dem Kunden wiederum weitergehende Services garantiert (Reaktionszeiten bei Fehlern, Updates, Versionen für bestimmte Plattformen o.Ä.). Als ein **Beispiel** unter vielen sei hier ***ownCloud*** genannt, das zwar unter der GPL entwickelt wird, dass v.a.

Geschäftskunden **parallel dazu aber eine "Business Edition"** unter einer proprietären Lizenz anbietet.

Nehmen wir ownCloud auch als Beispiel für ein weiteres verbreitetes **FLOSS-Business-Modell: Professional Services**. Vielen Unternehmen fehlt das notwendige technische Wissen inhouse, um eine komplexe und kritische Software verlässlich und v.a. sicher aufzusetzen und zu betreiben. Darum kaufen sie **professionelle Dienstleistungen** wie **Beratung**, **Maintenance** oder **Helpdesk** direkt vom Hersteller. Bei dieser Entscheidung spielen auch haftungsrechtliche Fragen eine Rolle, indem die Firma Risiken des Betriebs auf den Hersteller überträgt.

Vor allem für webbasierte Technologien ist ***Software as Service*** ein weiteres Geschäftsmodell. Hier betreibt ein Cloud-Anbieter eine Software wie ein Customer Relationship Management (CRM) oder ein Content Management System (CMS) auf seinen Servern und gewährt seinen Kunden Zugriff auf die installierte Anwendung. **Dies erspart dem Kunden Installation und Wartung der Software**, dafür zahlt er für die Nutzung der Software gemäß verschiedenen Parametern, etwa der Anzahl der Nutzer. Verfügbarkeit und Sicherheit spielen als unternehmenskritische Faktoren dabei eine große Rolle.

Zuletzt sei das vor allem bei kleineren Projekten häufige Modell genannt, zu einer freien Sofware **per Auftrag kundenspezifische Erweiterungen zu entwickeln**. Es obliegt dann meist dem Kunden, wie er mit diesen Erweiterungen verfährt, ob er sie also ebenfalls freigibt oder als Teil seines eigenen Geschäftsmodells unter Verschluss hält.

## Geführte Übungen

1. Wie lauten — in Kurzfrom — die "vier Freiheiten", wie sie von Richard Stallman und der Free Software Foundation definiert wurden?

| | |
|---|---|
| Freiheit 0 | |
| Freiheit 1 | |
| Freiheit 2 | |
| Freiheit 3 | |

2. Wofür steht die Abkürzung FLOSS?
3. Sie haben Software entwickelt und möchten sicherstellen, dass die Software selbst, aber auch alle künftigen darauf aufbauenden Werke ebenso frei bleiben. Welche Lizenz wählen Sie?

| | |
|---|---|
| CC BY | |
| GPL v3 | |
| 2-Clause BSD License | |
| LGPL | |

4. Welche der folgenden Lizenzen würde man als permissive bezeichnen, welche als Copyleft?

| | |
|---|---|
| Simplified BSD License | |

| | |
|---|---|
| GPL v3 | |
| CC BY | |
| CC BY-SA | |

5. Sie haben eine Webapplikation geschrieben und unter einer freien Lizenz veröffentlicht. Wie können Sie mit Ihrem Produkt Geld verdienen? Nennen Sie drei Möglichkeiten.

## Offene Übungen

1. Unter welcher Lizenz (einschließlich Version) stehen die folgenden Anwendungen?

| | |
|---|---|
| Apache HTTP Server | |
| MySQL Community Server | |
| Wikipedia articles | |
| Mozilla Firefox | |
| GIMP | |

2. Sie möchten Ihre Software unter der GNU GPL v3 veröffentlichen. Welche Schritte sollten Sie beachten?
3. Sie haben proprietäre Software geschrieben und würden sie gerne mit freier Software unter der GPL v3 kombinieren. Dürfen Sie das bzw. was müssen Sie beachten?
4. Warum hat die Free Software Foundation als Ergänzung zur GNU GPL die *GNU Affero General Public License 3* (GNU AGPL) veröffentlicht?
5. Nennen Sie drei Beispiele freier Software, die auch als “Business Edition”, also in einer kostenpflichtigen Version angeboten werden.

## Zusammenfassung

In dieser Lektion haben Sie gelernt:

- Gemeinsamkeiten und Unterschiede zwischen freier und Open Source Software (FLOSS)
- FLOSS-Lizenzen, deren Wichtigkeit und Probleme
- Copyleft vs. permissive Lizenzen
- FLOSS-Geschäftsmodelle