Bitkom-Leitfaden zu Open-Source-Software 2.0

(Gesamtentwurf)

Inhalt

Danksagung

Der vorliegende Leitfaden zu Open-Source-Software (OSS) ist eine gemeinsame Publikation der BITKOM-Arbeitskreise „ITK- Vertrags- und Rechtsgestaltung“ und „Open Source“. Die Arbeitskreise bestehen aus Experten der BITKOM-Mitgliedsunternehmen, die sich in ihrer beruflichen Tätigkeit intensiv mit dem Einsatz von OSS im Unternehmensumfeld beschäftigen. Die Erstellung dieses Leitfadens beruht jedoch weitgehend auf ihrem persönlichen ehrenamtlichen Engagement.

Für die Mitarbeit an diesem Leitfaden danken wir herzlich folgenden Personen, die mit ihrer Expertise und wertvollen praktischen Erfahrung die Erstellung der vorliegenden Publikation möglich gemacht haben:

Dr. Oliver Block, Bundesdruckerei GmbH

Antonia Byrne, Capgemini Deutschland Holding GmbH

Oliver Fendt, Siemens AG

Christine Forster, DATEV eG

Sigrid Freidinger, Nokia Solutions and Networks GmbH & Co. KG

Dr. Martin Greßlin, SKW Schwarz Rechtsanwälte

Björn Hajek, LL.M. (University College London), Infineon Technologies AG

Hans-Ulrich Hermann, Océ Printing Systems GmbH & Co. KG

Dr. Michael Jäger, Siemens AG

Sylvia F. Jakob LL.M. (Edinburgh), Institute for Legal Informatics, Leibniz Universität Hannover

Katharina Komarnicki, Siemens AG

Claudia Krell, SerNet GmbH

Marco Lechner, Accenture GmbH

Dr. Johannes Loxen, SerNet GmbH

Felix Mannewitz, Siemens AG

Karsten Reincke, Deutsche Telekom AG

Monika Schnizer, Fujitsu Technology Solutions GmbH

Dr. Hendrik Schöttle, Osborne Clarke

Martin Schweinoch, SKW Schwarz Rechtsanwälte

Sonja Schwuchow, LL.M., Kabel Deutschland Vertrieb und Service GmbH

Udo Steger, Unify GmbH & Co. KG

Axel Teichert, DB Systel GmbH

Dr. Christian-David Wagner, Wagner Rechtsanwälte

Dr. Hans Peter Wiesemann, DLA Piper UK LLP

Leser, die lediglich eine Einführung in die rechtlichen Aspekte von Open-Source-Software suchen, seien auch auf den nach wie vor einschlägigen BITKOM-Leitfaden zu Open-Source-Software 1.0 hingewiesen (http://www.bitkom.org/de/publikationen/38337_39870.aspx). Dabei handelt es sich um die Vorgängerversion dieses Leitfadens, die sich auf juristische Aspekte konzentriert.

Berlin, … (Datum der Fertigstellung des Leitfadens)

Executive Summary

Kernerkenntnisse, Merksätze, Handlungsempfehlungen

Das Executive Summary soll als letzter Schritt nach Fertigstellung der übrigen Kapitel erstellt und beschlossen werden.

1. Einleitung

Die erste Version des BITKOM-Leitfadens zum Thema Open-Source-Software (OSS) erschien im Jahr 2006 – und liegt damit schon einige Jahre zurück. Trotzdem hat sich seitdem wenig an einem grundsätzlichen Parameter geändert: Open-Source-Software (OSS) gewinnt im Unternehmensumfeld fortlaufend an Bedeutung. Sie einzusetzen, zu verwerten, zu bearbeiten und zu vervielfältigen, ist für viele zur Selbstverständlichkeit geworden. Mittlerweile sind die Grabenkämpfe um Vor- oder Nachteile einer geerdeten Aufgeklärtheit gewichen. Aus früheren Diskussionen bekannte Argumente wie „kostenlos“, „keine Beschränkungen“ auf der Seite der Befürworter und „keine professionelle Unterstützung“ oder „keine strukturierte Weiterentwicklung“ auf der Gegenseite werden heute deutlich differenzierter betrachtet. Stattdessen ist in der Post-Snowden-Ära beispielsweise der Sicherheitsaspekt verstärkt in den Mittelpunkt gerückt.

Immer relevant sind die rechtlichen Grundlagen für Herstellung und Einsatz von OSS und damit auch Lizenzfragen. Beliebtheit und zunehmende Verwendung stehen nach wie vor in Kontrast zu einer grundsätzlichen Unkenntnis: So selbstverständlich die Verwendung ist, so verbreitet sind oft auch Wissenslücken über Grundanforderungen im Umgang mit OSS, sei es als Basis für einen erfolgreichen Einsatz im Alltag oder als Voraussetzung für die Verwendung für eigene Softwareprojekte. Ein gängiges Missverständnis besteht z.B. darin, dass aus der einfachen und unentgeltlichen Verfügbarkeit von OSS auf das Fehlen jeglicher juristischen Einbettung geschlossen wird. Dass auch Open-Source-Software regelmäßig Nutzungsbedingungen unterliegt, wird übersehen oder schlichtweg ignoriert. Nachlässigkeiten in Sachen Open-Source-Software sind oft eine zweiseitige Angelegenheit, etwa, wenn einerseits OSS-Autoren z.B. die Markenrechte von Dritten nicht beachten, oder andererseits Nutzer die Rechte von OSS-Autoren bei GPL-Verletzungen ignorieren.

Die Neuauflage dieses BITKOM-Leitfadens betrachtet Open-Source-Software unter verschiedenen Aspekten – immer mit dem Ziel, Unternehmen einen ersten Überblick über die zahlreichen, unternehmensrelevanten Punkte zu verschaffen. Um diese adäquat abdecken zu können, wurde die Publikation gemeinsam von den BITKOM-Arbeitskreisen „ITK- Vertrags- und Rechtsgestaltung“ und „Open Source“ erarbeitet.

Kapitel 2 bietet den Einstieg, um sich mit **Begriffen und Grundlagen** vertraut zu machen. Kapitel 3 und 4 gehen vor allem auf **wirtschaftliche Aspekte** ein, stellen **Geschäftsmodelle** und die **Entstehung von OSS** vor. Um Zertifikate für Lizenzen, Verträge und Schutzrechtsverletzungen soll es unter anderem in Kapitel 5 **„Standardisierung und Kundenschutz“** gehen.

Kapitel 6 baut auf dem BITKOM-Leitfaden „Open-Source-Software – Rechtliche Grundlagen und Hinweise (Version 1.0)“ aus dem Jahr 2006 auf. Das Kapitel aktualisiert die **rechtlichen Anforderungen** beim Umgang mit OSS. Es soll ein Bewusstsein für die rechtliche Situation schaffen und Grundanforderungen bei der Verwendung von Open-Source-Software skizzieren, die ein Unternehmen in jedem Fall bedenken müsste. Mit der Beachtung dieser Grundanforderungen hat ein Unternehmen noch nicht jedes rechtliches Risiko ausgeschaltet, aber wichtige Grundlagen für ein rechtskonformes Verhalten geschaffen.

Den Wechsel zur **Anwender-Perspektive** vollzieht schließlich Kapitel 7. Dieses nimmt sich der Fragen an, wie mit Open-Source-Software im Unternehmen umzugehen ist: Welche Vorkehrungen sind z.B. in Bezug auf Strategie und Lizenzmanagement zu treffen?

Der Leitfaden schließt mit einer Aufzählung von **Risiken und Chancen** von Open-Source-Software in Kapitel 8. Dieses Kapitel verfolgt durchaus das Ziel, eine weiterführende Diskussion anzustoßen.

Dieser Leitfaden enthält gezielte Hinweise für:

- Unternehmen, die mit der Herstellung oder Lieferung von OSS Geld verdienen
- Unternehmen, die proprietäre Hard- oder Software herstellen und im Rahmen dessen in aller Regel (bewusst oder unbewusst) OSS-Komponenten einsetzen
- Unternehmen, die Software für die eigene Nutzung oder den Weitervertrieb beschaffen
- Unternehmen, die ihren Mitarbeitern die Zulieferung in OSS-Projekte erlauben oder sie sogar dafür bezahlen
- Unternehmen, die öffentliche Stellen bei der Ausschreibung von SW-Beschaffung beraten.

Die Publikation wendet sich sowohl an Juristen als auch an Nichtjuristen. Sie möchte Geschäftsführer kleiner und mittlerer Unternehmen ebenso unterstützen wie Mitarbeiter von Rechtsabteilungen oder die Experten aus Entwicklungs- und technischen Fachabteilungen der Unternehmen.

Dieser Leitfaden will eine erste Orientierungshilfe sein und versucht, bewährte, in der Praxis umsetzbare Anwendungshilfen für den Umgang mit Open-Source-Software zu geben. Die Materie ist allerdings sehr komplex und der fortlaufenden Entwicklung von Recht und Technik unterworfen. Daher erhebt dieser Leitfaden **keinen Anspruch auf Vollständigkeit**. Er kann **keine umfassende juristische Aufbereitung** leisten oder konkrete Hinweise für die Vertragsgestaltung geben. Die Einbindung professioneller unternehmensinterner oder -externer Experten und (rechtlicher) Berater ist in jedem Falle anzuraten.

Für interessierte Leser, die sich vertieft mit dem Thema auseinandersetzen möchten, finden sich weiterführende Literaturempfehlungen im Anhang.

2. Grundlagen

2.1 Einsatz von Open-Source-Software

Open-Source-Software (OSS) ist **weit verbreitet**. Jedes Android-Smartphone enthält sie. Das freie Betriebssystem GNU/Linux bildet die Basis vieler Computer – kleiner embedded-Rechner ebenso wie großer Desktop- und Servermaschinen; selbst Mainframesysteme setzen darauf. Apache, Tomcat, PostgreSQL oder MySQL sind gern genutzte Open-Source-Komponenten für Server. Die Softwareentwicklung an sich – unter C/C++, Java, PHP und vielen anderen (Skript-)Sprachen – wird ausgereift unterstützt. Eclipse ist ein sehr prominenter Vertreter entsprechender Tools. Außerdem stehen Nutzern komplette Officeprogramme zur Verfügung; Libre Office ist hierfür bekanntestes Beispiel. Für nahezu jedes spezifische Interesse ist eine freie Softwarelösung in hoher Qualität verfügbar – sei es die rechnergesteuerte Bild-, Ton-, und Filmbearbeitung oder das professionelle Layouten. Und mehr noch: Selbst in kommerziellen Produkten werden in immer stärkerem Maße OSS-Komponenten integriert, sei es als Tools, als Bibliotheken oder als Frameworks. Oder anders gesagt: Die Wahrscheinlichkeit, auf einem beliebigen Desktopcomputer oder einem Server OSS-Komponenten zu finden, ist heute sehr hoch. Open-Source-Software ist kein Exot mehr. Ihre Nutzung ist alltäglich und selbstverständlich.

Open-Source-Software gilt als **zuverlässig und sicher**. Für manche ist sie sogar ein Innovationsmotor mit ungeheurem Potenzial. Neben der **Vielfalt der Anwendungsszenarien** steht die **Fülle der Anwendungen**. Freie Software – ein gängiges anderes Etikett für Open-Source-Software – hat **echte wirtschaftliche Bedeutung** gewonnen. Die Serverinfrastruktur einiger großer Internetfirmen basiert zum größten Teil auf Linux. Google, Facebook oder Amazon sind bekennende Nutzer. Die Gründe dafür sind simpel: Unternehmen müssen für den Einsatz von Open-Source-Software keine Lizenzkosten einkalkulieren. Das vereinfacht den Aufbau skalierender Umgebungen signifikant. Außerdem können die Unternehmen die genutzte freie Software im Rahmen der jeweiligen Lizenzbestimmungen entsprechend eigener Zwecke und Ziele weiterentwickeln. Selbst wenn die modifizierte Version danach geschäftskritische Elemente enthält – und der wirklich geschäftskritische Anteil eines Softwaresystems ist in der Regel sehr klein –, gibt es immer noch gute Einsatzszenarien, die deren Schutz in Verbindung mit Open-Source-Software gewährleisten.

Auch Hersteller, die Internetrouter und andere **Hardwaresysteme** in großen Stückzahlen produzieren, nutzen heute fast ausschließlich Linux. Die kommerziellen Unix-Formate Solaris (SUN), AIX (IBM) und HP-UX (HP) wurden in spezialisierte Nischen zurückgedrängt. Linux wird von diesen Herstellern inzwischen oft sogar stärker unterstützt als das eigene Unix. Denn jedes Unternehmen kann so wechselseitig auf alle Innovationen in Linux zugreifen – gerade darin zeichnet sich das Open-Source-Wesen aus, das Kooperation und Kollaboration sowie das Teilen der Ergebnisse hochhält: Technologien, die in einem offenen Prozess entwickelt werden, stehen allen zur Verfügung. Konsequenterweise muss ein Unternehmen das komplexe Ganze nicht mehr eigenständig im jeweils eigenen Unix aufbauen und pflegen. Stattdessen gibt jeder ein wenig und erhält in der Summe das Ganze. Die Kosteneinsparung auf der Entwicklungsseite ist damit evident und der Wettbewerb verlagert sich in Richtung strategischer Spezialisierungen und Serviceangebote.

Diese Hinwendung zu einer gemeinsamen, kooperativen Bereitstellung der Technologie an sich und die Konzentration auf ein Alleinstellungsmerkmal, das sich in einem auf dieser Technologie aufbauenden Service manifestiert, zeigt sich aktuell bei Cloudservices: OpenStack und Docker sind prominente Technologien, auf denen unterschiedliche Firmen aufbauen und mit denen sie konkurrierende Angebote in den Markt bringen – obwohl sie zugleich die Technologie erfolgreich gemeinsam weiter entwickeln.

Das macht deutlich, dass man mit **Open-Source-Software** sehr wohl **Gewinn** erzielen darf. Dass OSS nicht kommerziell verwendet werden dürfe, ist ein Irrglaube. Das Gegenteil ist der Fall: man darf auf der Grundlage von Open-Source-Software sogar ganze Geschäftsmodelle entwickeln.

Allerdings können diese Geschäftsmodelle nicht darin bestehen, über Lizenzkosten und Lizenzverträge Umsatz zu generieren. Denn auch im kommerziellen Kontext bleibt das Wesen der

Open-Source-Software natürlich erhalten: konstitutiv gehört hierzu, dass für den Kunden **keine Lizenzkosten** anfallen dürfen. Für die aufgewandte Arbeit für die Zusammenstellung und die Installation von Software als ein Service darf allerdings ebenso ein Entgelt verlangt werden wie für generelle Leistungen, die mit dem Einsatz von Open-Source-Software zusammenhängen. Ausgeschlossen ist eben nur, dass der Servicegeber seinen Kunden die bloße Nutzung der Software in Rechnung stellt. Alles andere darf er sich vergüten lassen. Oder anders: Mit Open-Source-Software verlagert sich das Geschäftsmodell notwendig weg vom Lizenzgeschäft hin zur Subskription von Diensten mit und um Open-Source-Software herum.

Dennoch ist der Einsatz von Open-Source-Software nicht zum absoluten Nulltarif zu bekommen. Auch wenn die Verwendung bzw. das Recht dazu – pekuniär gesehen – in der Tat kostenlos ist, und zwar immer. Es gehört zur Definition von Open-Source-Software, dass reine Lizenzkosten für das Recht zur Nutzung von Open Source Software nicht anfallen dürfen. Unter bestimmten Umständen muss der Anwender allerdings eben doch aktiv und auf eigene Kosten Bedingungen erfüllen, die an die Softwarenutzung gekoppelt sind. OSS folgt damit dem Prinzip **„Paying by Doing“**, selbst wenn die Kosten für dieses „Tun“ zumeist zu vernachlässigen sind.

2.2 Konzept und Definition von Open-Source-Software

Um Open-Source-Software effektiv einzusetzen, ist es hilfreich, zunächst das Konzept „Open-Source-Software“ selbst zu verstehen, das sich auch in und mit einem gewissen Begriffsapparat manifestiert. Es gibt eine **Kernidee**, die jede Software erfüllen muss, wenn sie freie Software sein will: wer immer sie hat, muss auch das **Recht** haben, sie (a) **auszuführen**, sie (b) zu **analysieren,** sie (c) an die eigenen Bedürfnisse **anzupassen** und sie (d) **weiterzugeben**, selbst in veränderter Form.[1] Diese vier Kernrechte sind konstitutive und unverzichtbare Bestandteile sowohl der Definition für Open-Source-Software als auch der Definition für freie Software; insofern ist Open-Source-Software auch freie Software. Das namensgebende Kriterium, dass bei freier Software oder bei Open-Source-Software auch der Quellcode offen zugänglich bereitstehen muss, ist somit eine notwendige Voraussetzung für die Klassifizierung, jedoch keine hinreichende[2]. Allerdings setzt die Ausübung der vier Kernrechte den Zugriff auf den Quellcode voraus.

Die für Open-Source-Software **konstitutiven Rechte** der **Nutzung**, **Einsicht**, **Veränderung** und **Distribution** (auch in veränderter Form) sind nicht per se mit Software verbunden: Software unterliegt dem Urheberrecht.[3] Danach liegen sämtliche Rechte an einer Software zunächst bei deren **Urheber**. Nur die Urheber können bestimmen, was andere mit dem von ihnen geschaffenen Werk tun dürfen und was nicht. Damit andere Personen als der Urheber Software in rechtmäßiger Weise einsetzen können, muss ihnen der Urheber eine **Nutzungsberechtigung** einräumen. Für eine solche Nutzungsberechtigung, die regelmäßig mit bestimmten Nutzungsbedingungen und / oder Nutzungsbeschränkungen verbunden ist, hat sich der Begriff „**Lizenz**“ eingebürgert. Will also ein Urheber – oft der Programmierer oder sein Arbeitgeber – eine Software als Open-Source-Software vertreiben, muss er die genannten Freiheitsrechte explizit einräumen. Er muss sein Werk unter eine freie Lizenz stellen, unter eine Open-Source-Lizenz.

Historisch gesehen gibt es viele Software-Lizenzen, die oben genannte Rechte mit Software verknüpfen. Um eine gewisse Ordnung in die Vielfalt der Lizenzmodelle zu bringen, hat sich die Open Source Initiative (OSI) gegründet.[4] Sie hat existierende Lizenzvorgaben gesammelt, gesichtet und zur zehn Kriterien umfassenden **Open-Source-Definition** (OSD) kondensiert.[5] In der Folge ist Open-

[1] vgl. die Begriffsklärung der Free Software Foundation Europe zu freier Software (URL: http://fsfe.org/about/basics/freesoftware.de.html)
[2] Eine noch weiter reichende Differenzierung von Free- zu Open-Source-Software bietet u.a. die http://www.gnu.org/philosophy/free-sw.html
[3] vgl. z.B. für Deutschland §§ 69a ff. des Urhebergesetzes (UrhG)
[4] vgl. http://opensource.org/
[5] vgl. http://opensource.org/osd

Source-Software nur dann wirklich Open-Source-Software, wenn sie unter einer Lizenz freigegeben worden ist, welche die zehn Kriterien der OSD erfüllt. Die Kriterien der OSD schließen bruchlos die vier **Merkmale von freier Software** ein, welche die Free Software Foundation postuliert.[6] **Deshalb ist jede Open-Source-Software auch freie Software, nicht aber umgekehrt.** Die Unterscheidung von Free Software und Open-Source-Software ist aus rechtlicher Sicht – und damit für einen weiten Teil dieses Leitfadens – nicht wirklich relevant. Für die rechtliche Beurteilung kommt es allein auf die Nutzungsbedingungen an, also auf die konkrete Open-Source-Lizenz.[7]

Die OSI ist außerdem über die Festlegung der Definition hinausgegangen und hat einen „**Licence Review Process**" mit dem Ziel eines „Approvals" implementiert.[8] Damit unterliegt es nicht der persönlichen Interpretation, ob es sich im Einzelfall wirklich um Open-Source-Software bzw. um eine Open-Source-Lizenz handelt. Für über 60 Open-Source-Lizenzen hat die OSI sichergestellt, dass sie die Kriterien der Open-Source-Definition erfüllen und listet sie offiziell.[9] Dies hat einen praktischen Wert: Wer Software einsetzt, die unter einer offiziell gelisteten Lizenz freigegeben worden ist, weiß auch ohne Blick in die einzelne Lizenz, dass er im Hinblick auf diese Software alle Rechte hat, die die OSD zur Voraussetzung macht. Der Anwender weiß außerdem, dass er keine Einschränkungen berücksichtigen muss, die die OSD ausschließt. Dies gilt allerdings nur, soweit es sich um einen von der OSI freigegebenen, unveränderten Lizenztext handelt. Welche Bedingungen allerdings an die Ausübung der Rechte aus einer Lizenz gekoppelt sind, offenbart letztlich nur die jeweilige Lizenz selbst. Jedenfalls ist Open-Source-Software letztlich nur dann wirklich Open-Source-Software, wenn sie unter einer offiziellen Open-Source-Lizenz freigegeben worden ist, die die zehn Kriterien der OSD erfüllt und die von der OSI bestätigt ist.

Noch offen ist an dieser Stelle die Frage, wie Software zu bezeichnen ist, die nicht Open-Source-Software ist. Geläufig sind die Begriffe „**proprietäre" oder „kommerzielle Software**". Sie sind jedoch ungenau. So hat auch Open-Source-Software Urheber und Rechteinhaber. Und auch Open-Source-Software kann kommerziell eingesetzt werden. Es hat sich darum der Begriff „**Closed Software**" als begriffliches Gegenstück etabliert. Man spricht von „closed", also „geschlossener" Software, weil bestimmte Freiheitsrechte verschlossen bleiben (unter anderem, aber nicht nur bleibt der Quellcode unzugänglich). So erschließt sich auch das „offen" von Open-Source-Software nicht allein aus dem offen vorliegenden Quellcode, sondern weil sie das Recht zu (u.a.) Einsatz, Veränderung und Weitergabe auch in veränderter Form eröffnet. Eine weitere Gegenüberstellung entsteht durch die Analyse von Open-Source-Lizenzen unter dem Aspekt des Patentrechts.[10] Denn eine Patentierung entzieht einer Software die freie und unentgeltliche Nutzbarkeit.

Auch wenn die rechtliche Beurteilung einer Software nicht von der Benennung, sondern von den jeweiligen Nutzungsbedingungen abhängt, hat sich trotzdem ein Diskurs über Lizenztypen etabliert. Die Begriffe „freie Software" und „Open Source Software" sind solche verallgemeinernden Begriffe. Und man wird in der Praxis auch einer gewissen Laxheit begegnen. Nicht alles, was als Open-Source-Software bezeichnet wird, ist formal auch Open-Source-Software. Will man angemessen am Diskurs teilnehmen, sollte man den Terminus „Open-Source-Software" nur verwenden, wenn die in Rede stehende Software zumindest die vier Kriterien der Free Software Foundation erfüllt. Eine missbräuchliche Falschbezeichnung als Open-Source-Software oder freie Software könnte als Verstoß gegen das Wettbewerbsrecht gewertet werden.

---

[6] vgl. http://www.gnu.org/philosophy/philosophy.html
[7] vgl. hierzu Kapitel 2.3 und 6.1
[8] vgl. Open Source Initiative: The License Review Process (URL: http://opensource.org/approval)
[9] vgl. Open Source Initiative: Licenses by Name (URL: http://opensource.org/licenses/alphabetical)
[10] vgl. Chapter 3.1 „The problem of implicitly releasing patents"; in: Reincke, Karsten / Sharpe, Greg. Open Source License Compendium – How to Achieve Open Source License Compliance. (URL: http://opensource.telekom.net/oslic/releases/oslic.pdf)

2.3 Lizenz-Typen von Open-Source-Software

Open-Source-Software tritt in verschiedenen Formen in Erscheinung, deren Ausprägungen für den Einsatz der Software im konkreten Fall entscheidende Bedeutung haben können. Zwar erfüllen alle von der OSI anerkannten **Open-Source-Lizenzen** alle Kriterien der Open-Source-Definition. Dennoch **unterscheiden** sie sich. Denn es steht dem Lizenzgeber frei, zusätzliche Rechte einzuräumen. Vor allem dürfen die Open-Source-Lizenzen von den Anwendern der so lizenzierten Software die Erledigung recht unterschiedlicher Aufgaben fordern. Das reicht von der einfachen Nennung der genutzten Software und ihres Schöpfers bis zu differenzierten Konsequenzen für das übergeordnete Programm oder System. Der Katalog möglicher Forderungen mag komplex erscheinen – die Erfahrung zeigt jedoch, dass die im konkreten Einzelfall zu erbringende Gegenleistung zumindest pekuniär zu vergleichsweise günstigen Konditionen umgesetzt werden kann. Ein wenig teurer ist es vielleicht, das Wissen zu erwerben, was genau unter welchen Umständen zu tun ist. Die reinen Kosten für die Umsetzung reduzieren sich jedoch in der Regel auf eine vernachlässigbare Größe – nur die Umsetzung selbst darf eben nicht vernachlässigt werden, wenn man die Software lizenzkonform und legal nutzen will.

Die Gruppierung in den folgenden Abschnitten erleichtert das grundlegende Verständnis von Open-Source-Lizenzen und die mit ihnen verknüpften Implikationen.[11]

2.3.1 Der Copyleft-Effekt

Ein besonders wichtiges Kriterium für die Kategorisierung und für das Verständnis von Open-Source-Lizenzen ist der sogenannte „Copyleft-Effekt“. Erfunden wurde der Begriff „Copyleft“ von Richard Stallman als Wortspiel. Seiner Meinung nach nutzen Copyright-Owner das Copy-RIGHT, um den Nutzern von Software Rechte vorzuenthalten, die ihnen gehören sollten. Stallman wollte hingegen sicherstellen, dass diese Rechte den Nutzern nicht genommen werden können. Copy-LEFT bezeichne deshalb eine Methode, „[...] ein Programm (oder anderes Werk) frei zu machen und zu verlangen, dass alle modifizierten und erweiterten Programmversionen ebenfalls frei sind“. Der Urheber lizenziert seinen Code so, dass nicht nur sein eigener Code frei genutzt, frei geändert und frei weitergegeben werden darf, sondern dass auch alle **Änderungen, Ergänzungen und Ableitungen** im gleichen Sinn **frei genutzt werden dürfen**, die andere Entwickler in den Entwicklungsprozess einbringen.[12] Verkürzt gesagt, meint „Copyleft“ also, dass man seine Bearbeitungen nur unter denselben Bedingungen weitergeben darf, unter denen man den Ausgangsstand seiner Bearbeitungen erhalten hat.

In der Praxis ergibt sich außerdem die Notwendigkeit, zwischen **starkem** und **schwachem Copyleft** zu unterscheiden: Das starke Copyleft will dafür sorgen, dass die Software, die ein so lizenziertes Werk als eine konstitutive Komponente verwendet, unter denselben Bedingungen weitergegeben wird, unter denen man die Komponente erhalten hat. Die ursprünglichen Lizenzbedingungen erstrecken sich also auch auf Änderungen, Hinzufügungen, etc. Das schwache Copyleft möchte dies nur für das übernommene Werk und dessen direkte Änderungen sicherstellen. Es gibt bei Lizenzen mit schwachem Copyleft ein Abgrenzungskriterium, welches einen Code sozusagen „in zwei Teile teilt“. So wird es möglich, zu einer unter schwachem Copyleft stehenden Bibliothek im Rahmen eines abgeleiteten Werkes hinzugefügten, unabhängigen Code bei Beachtung des Abgrenzungskriteriums unter eine andere Lizenz zu stellen. Für diese andere Lizenz gelten weniger oder keine Einschränkungen. Nur auf der Copyleft- oder Bibliotheksseite des Abgrenzungskriteriums kommen die viel stärkeren Copyleft-Regelungen zur vollen Geltung.

---

[11] Neben der folgenden Differenzierung von Copyleft- und permissiven Lizenzen entsteht eine zweite Art der Gruppierung, wenn man die Open-Source-Lizenzen hinsichtlich ihres Umgangs mit Softwarepatenten analysiert. Näheres dazu in Chapter 3.1 „The problem of implicitly releasing patents“, in: Reincke, Karsten / Sharpe, Greg. Open Source License Compendium – How to Achieve Open Source License Compliance. (URL: http://opensource.telekom.net/oslic/releases/oslic.pdf)

[12] vgl. GNU-Projekt (URL: http://www.gnu.org/copyleft/copyleft.de.html)

Die Idee des Copyleft-Effekts ist nicht wirklich intuitiv zugänglich. Eine Alltagsanalogie möge das Verständnis erleichtern:

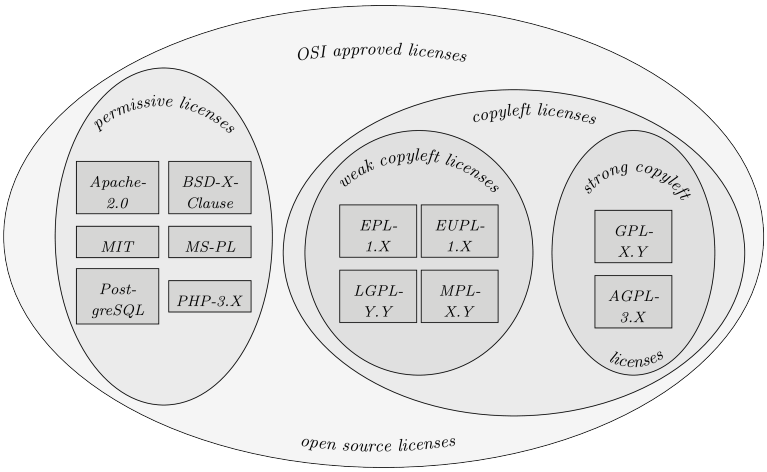
Man begegnet der traditionellen „Weitergabe“ von zubereiteten Mahlzeiten oft genug. In jedem Restaurant, in jeder Kantine, bei vielen Einladungen. Eine quelloffene Verteilung von Mahlzeiten läge bei dieser Analogie dann vor, wenn nicht nur das Essen zum Verzehr zur Verfügung gestellt würde, sondern auch die entsprechenden Rezepte.

Was der Restaurantkunde im Einzelnen mit den Rezepten tun dürfte, hinge natürlich von den Bedingungen ab, unter denen ihm das Rezept übergeben wird. Dürfte er sie nachkochen, das Rezept selbst verfeinern und das verfeinerte Ergebnis auch an andere Gäste und Köche weiterreichen, läge sozusagen „freie Nahrung“ vor.

Und damit ergibt sich der Copyleft-Effekt nahezu zwanglos: Gesetzt, jemand böte eine Suppe als „freie Nahrung“ an, also die Suppe an sich und das Rezept. Wenn seine Regeln für die Verwendung des Rezepts dann festlegen würden, dass jede Variante dieses Suppenrezeptes selbst nur als „freie Nahrung“ verteilt werden dürfte, dann hätten seine Nutzungsregeln ein schwaches Copyleft. Die Wirkung dieser Regeln wäre „schwach“, weil sie nur erzwingen würden, dass alle gekochten Suppen, die Varianten des Ausgangsrezeptes realisieren, als „freie Nahrung“ weitergegeben werden müssten. Lägen die Regeln zur Nutzung der Rezepte dagegen fest, dass auch das ganze Menü, in das eine solche Suppe als ein Gang eingebaut würde, als „freie Nahrung“ angeboten werden müsste, dann würden diese Regeln ein starkes Copyleft etablieren. Es ist stark, weil es über die Suppe hinaus auch das ganze Menü erfasst.

So lässt sich sofort auch die dritte Klasse von „Rezeptnutzungsregeln“ verdeutlichen. Überließen die Regeln es dem Empfänger von „freier Nahrung“, ob er seine eigenen Varianten der Suppe auch wieder als „freie Nahrung“ an Dritte weitergäbe oder ob er sie „traditionell“, also ohne Rezept zum Verzehr anböte, dann läge eine permissive Nutzungsregelung vor.

Und damit zurück in die Welt der Software: Hier spricht man von permissiven Open-Source-Software-Lizenzen, von OSS-Lizenzen mit schwachem Copyleft und von OSS-Lizenzen mit starkem Copyleft.[13]



[13] vgl.: Reincke and Sharpe: Open Source License Compendium, Release 1.0, 2015, p.22 (URL: http://opensource.telekom.net/oslic/releases/oslic.pdf)

2.3.2 Permissive Lizenzen

Permissive Lizenzen sind die dritte Untergruppe der Open-Source-Lizenzen. Sie haben **keinen Copyleft-Effekt** – weder einen schwachen noch einen starken. Diese Lizenzen überlassen es dem Bearbeiter der Software, ob er seine eigene übergreifende Arbeit oder seine Änderungen an dem übernommenen Werk auch wieder als freie Software weitergibt. **Sie stellen ihm frei, das Ergebnis seiner Bearbeitung zu Closed Software zu machen**, einfach, indem sie es ihm überlassen, ob er den Quellcode weitergibt oder nicht. Eben dieses Anheimstellens wegen hat sich für diese Art der Open-Source-Lizenzen auch der Begriff der permissiven Lizenzen eingebürgert.

Sie erlauben dem Bearbeiter, das Ergebnis seiner Verbesserungen und Veränderungen sozusagen für sich zu behalten, obwohl diese Arbeit auf einer freien Softwarebasis aufsetzt. Permissive Lizenzen lassen zu, dass der Bearbeiter das Ergebnis seiner Eingriffe als Closed Software in Binärform weitergeben kann. Unabhängig davon fordern auch die permissiven Lizenzen die Umsetzung bestimmter Aspekte, nur eben andere bzw. weniger.

Zur **Gruppe der permissiven Lizenzen** gehören mindestens:

- Apache-Lizenz (= Apache License, Version 2.0)[14]
- Berkeley Software Distribution Lizenzen[15]
  - BSD-2-Clause
  - BSD-3-Clause
- MIT-Lizenz (MIT License)
- MS-PL (Microsoft Public License)
- PostgreSQL-Lizenz (PostgreSQL)
- PHP-Lizenz (PHP License 3.0).

2.3.3 Lizenzen mit schwachem Copyleft

Softwarelizenzen mit schwachem Copyleft fordern – nebst all den anderen Bedingungen, die sie auch erfüllt sehen wollen –, dass Veränderungen und Verbesserungen unter denselben Bedingungen genutzt und weitergegeben werden wie die, unter denen der Bearbeiter das Original erhalten hat, also unter derselben Lizenz. Diese Bedingungen schließen ein, dass die Veränderungen der erhaltenen Software selbst auch wieder im Quellcode zugänglich gemacht werden müssen, soweit diese auf der Copyleft-Seite des Abgrenzungskriteriums liegen.[16] Sie dehnen ihren Anspruch aber nicht auf die Software aus, die eine Software mit schwachem Copyleft als eingebettete Komponente nutzt.

Zur Gruppe der Lizenzen mit schwachem Copyleft gehören mindestens:

- EPL (= Eclipse Public License 1.0)[17]
- EUPL (= European Union Public License, Version 1.1)[18]
- LGPL (= GNU Lesser General Public License, Version 2.1 / 3.0)
- MPL (= Mozilla Public License, Version 1.1 / 2.0)
- MS-RL (= Microsoft Reciprocal License).

---

[14] Hinweis zu den Bezeichnungen: Die Nomenklatur der Lizenzen folgt soweit möglich der SPDX License List (URL: http://spdx.org/licenses/), die das Ziel einer einfachen und effizienten Wiederauffindbarkeit von Lizenzen durch einheitliche Bezeichnungen und Identifier verfolgt.

[15] BSD-Lizenzen verlangen eine erhöhte Aufmerksamkeit, da sie auf die Idee eines Templates hin angelegt sind: Man übernimmt den Lizenztext, spezifiziert die Copyright-Line und nennt das Ganze seine XYZ-Lizenz. Bei der BSD-Lizenz ist es also notwendig, den Lizenztext selbst anzusehen, um festzustellen, ob es wirklich eine BSD Lizenz ist. In den anderen Fällen wird ein solch vertiefender Blick auch nicht schaden.

[16] siehe oben unter Ziffer 2.3.1

[17] Die Einordnung der EPL ist umstritten, teilweise wird sie als Lizenz mit starkem Copyleft gesehen.

[18] Die Einordnung der EUPL ist umstritten, teilweise wird sie als Lizenz mit starkem Copyleft gesehen.

2.3.4 Lizenzen mit starkem Copyleft

Lizenzen mit starkem Copyleft fordern – nebst all den anderen Bedingungen, die sie auch erfüllt sehen wollen –, dass Veränderungen und Verbesserungen unter denselben Bedingungen genutzt und weitergegeben werden wie die, unter denen der Bearbeiter das Original erhalten hat, und zwar die Veränderungen und Verbesserungen der Originalsoftware an sich und die übergreifende Software, die das Original als Subkomponente benutzt. Diese Bedingungen schließen ein, dass die Veränderungen der erhaltenen Software ebenso im Quellcode zugänglich gemacht werden, wie das auf dem Original oder seiner veränderten Version aufsetzende neue Ganze.

Zur Gruppe der Lizenzen mit starkem Copyleft gehören mindestens:

- GPL (= GNU General Public License, Version 2.0 / 3.0)
- AGPL (= GNU Affero General Public License, Version 3.0).

Die Feinheiten in der Unterscheidung von Lizenzen dürfen aber von einem nicht ablenken: Für sie alle gilt das Paying-by-Doing-Prinzip. Selbst die permissivste Lizenz erwartet vom Nutzer einer so lizenzierten Software, dass er unter bestimmten Umständen etwas tut. Um also Open-Source-Software lizenzkonform zu nutzen, reicht es nicht, in generalisierten Kategorien zu denken. Nur die konkrete Lizenz legt fest, was genau unter welchen Umständen zu tun ist. Am Ende geht es um den Einzelfall, um das, was die spezifische Open-Source-Lizenz konkret einfordert.

2.4 Zur Entstehung von Open-Source-Software

Open-Source-Software hat auch als Idee eine Geschichte.[19] Sie in Grundzügen zu kennen, erleichtert das Verstehen - und dieser Aufgabe widmen sich die folgenden Abschnitte.

2.4.1 Über Unix zu Linux

Der Begriff „Open-Source-Software“ selbst ist eine recht junge Erfindung. Erst 1998 – sozusagen als „Marketingkniff“ – erschaffen, sollte er den bis dato gebräuchlichen Terminus „freie Software“ ersetzen, der sich seit den frühen 1980er Jahren etabliert hatte. Mit dem Begriff „Free Software“ setzte (und setzt) man auch politische Signale. So entstand in der Community der Wunsch nach einer pragmatischen, nach einer unverfänglichen Alternative.

Abseits dieser Programmatik war freie Software jedoch **seit den Anfängen** der kommerziellen Computertechnik präsent: In den ersten Jahrzehnten bildete freie Software bei Computerfirmen eine konstitutive Komponente ihres Geschäftsmodells, ohne allerdings schon als freie Software bezeichnet zu werden. Das eigentliche Geschäft der kommerziellen Informatik bestand anfänglich nämlich „nur“ im Verkauf von Hardware. Software war das „Give-Away“, das inklusive Quellcode zusammen mit der Hardware ausgeliefert wurde. Der Austausch von verbesserten Versionen der Software unter den Kunden wurde als Verbesserung des Hardwareproduktes gesehen. Begünstigt wurde dieser Umgang durch die Art der Kundschaft: Die frühen Computersysteme wurden vor allem an Universitäten beheimatet. Wissenschaftler und Techniker nutzten die mitgelieferte Software, passten sie – als Teil ihrer Arbeit – an den jeweiligen Bedarf an, gaben sie weiter, empfingen veränderte Versionen und setzen eigene Arbeiten darauf auf. Sie begegneten damit dem Mangel an Einsatzmöglichkeiten der teuren Hardware. Umgekehrt war es im Sinne der Firmen, dass die Software für ihre Hardware wuchs.

---

[19] Wird man gezwungen, ein Buch zu nennen, das der Interessierte lesen sollte, wenn er sich schnell und gründlich kundig machen will, wird man wahrscheinlich auf das Buch „Die Software-Rebellen“ von Glyn Moody verfallen. Darf man zwei nennen, wird wohl auf die von Sam Williams unter dem Titel „Free as in Freedom. Richard Stallman“s Crusade for Free Software“ veröffentlichte Biografie zu verweisen sein. Die nachfolgende Darstellung folgt – ohne einzelne Nachweise aber mit ausdrücklichem Dank – diesen beiden Werken. Ansonsten ist die Literatur mittlerweile nahezu unübersehbar gewachsen.

Je größer deren Funktionsumfang war, desto stärker der Anreiz für Institute und Organisationen, sie zu kaufen.

Ende der 1960er Jahre wendete sich das Blatt: IBM wurde durch eine Klage des US Department of Justice gezwungen, Hard- und Software zu entkoppeln. Software bekam einen eigenen Wert. Diese Entkoppelung legte den Grundstein für einen eigenen Markt. Betriebssysteme und Programme wurden zum relevanten Unterscheidungsmerkmal der neuen Geschäftssparte. Hardware lieferte „nur noch“ die reine Rechenleistung. Software wurde zur handelbaren Ware.

In dieser Zeit entstand in den Bell Laboratories von AT&T auch die erste UNIX-Version, vornehmlich entwickelt für den internen Eigenbedarf. An die Universitäten jedoch gab AT&T ihr Unix zum Selbstkostenpreis ab, und zwar samt Quellcode. Dass diese Codebasis problemlos zugänglich war, erhöhte zuerst das Benutzerinteresse. Und dieses ermunterte zu Weiterentwicklungen, genauso, wie – ironischerweise – der fehlende Support: AT&T selbst nämlich bot keinen Support für das Produkt an. Also galt es, sich selbst zu helfen, vornehmlich und am besten im Austausch mit Gleichgesinnten. So schlossen sich schnell lokale Anwender zusammen, tauschten über Vorversionen des Internets Lösungen und Verbesserungen aus und halfen bei Problemen. Über diese freie unorganisierte Kooperation in der akademischen Community erreichte UNIX zügig einen hohen Reifegrad.

Einen **ersten Schritt in Richtung Institutionalisierung** machte dann die Universität von Kalifornien in Berkeley. Sie fasste die entstehenden Verbesserungen zusammen, sammelte die neuen Unixtools und -applikationen ein und veröffentlichte alles zusammen kontinuierlich in ihrer Berkeley Software Distribution (BSD). AT&T auf der anderen Seite wandte sich mit kommerziellem Interesse der Aufgabe zu, stabile Releases zu vertreiben. Als es **1984** durch den Fall des Telefonmonopols zur Aufspaltung von AT&T kam, griff aber auch die Nachfolgefirma das Konzept einer aus sich heraus werthaltigen Software auf. Nutzer der Tools aus der BSD mussten nun auch eine AT&T-Quellcodelizenz erwerben, die immer teurer wurde. Die Kommerzialisierung von Unix schritt voran, nicht ohne eine Gegenbewegung ins Leben zu rufen. So führte die Einschränkung der freien Nutzung über die Einführung von Lizenzgebühren auch dazu, dass – nach einem entsprechenden Aufruf in einer konzertierten Aktion der BSD-Entwickler – verbliebener AT&T-Code vollständig aus dem BSD-Unix-Derivat entfernt wurde. Mit der Zeit entstand so ein wirkliches freies Unix. Wenn wir heute (neben Linux) auch auf OpenBSD, FreeBSD usw. als ein freies Unix zugreifen können, dann ist das dieser Entwicklung zu verdanken.

Ein zweiter und anderer Weg führte zu Linux:

Auch an der Universität von Richard M. Stallman – dem Labor für Künstliche Intelligenz des MIT mit seiner ursprünglichen „Hacker“- und „Sharing“-Kultur in Sachen Software – machte sich die Entwicklung der Kommerzialisierung und der Abschottung praktisch bemerkbar. Gern erzählt wird die Anekdote, nach der Richard Stallman einmal seinen Drucker an neue Schnittstellen anpassen wollte, die dafür notwendige Druckersoftware aber nicht im Quellcode dem Drucker beigegeben worden war. Selbst als Stallman endlich einen universitären Kollegen gefunden hatte, der diesen Code besaß, durfte ihn der Kollege aufgrund der Lizenzvereinbarung nicht an Richard Stallman weitergeben. Software hatte auf einmal einen eigenen Wert, der durch Abgrenzung geschützt werden musste. Und eben diese Ausgrenzung – so die einfache Sichtweise von Richard M. Stallman – behindere ihn in seinem Tun.

Diese Art des Schutzes durch Abkehr von der Idee einer freien Nutzung griff bald über das kommerzielle Betriebssystem Unix hinaus. Seit 1981 in den USA Patente auf Software möglich wurden, entstand auch unter den Entwicklern selbst vermehrt der Wunsch, Programme zu besitzen. Auch von dieser Seite aus schränkte sich also die Bereitschaft zum freien Austausch von Software ein.

Stallman empfand beide Entwicklungen als eine persönliche Beschränkung. Und so versuchte er ab **1984,** die ursprüngliche Dynamik einer Kultur des Teilens, wie er sie in der frühen UNIX- und BSD-Gemeinschaft im MIT kennen und schätzen gelernt hatte, wieder zu beleben. Er begann, das **Betriebssystem GNU** (ein rekursives Akronym: „GNU“s not UNIX“) unter den Bedingungen freier

Software zu entwickeln und zu vertreiben. Mehr noch: er konzipierte die Idee freier Software und wurde so zum Vorreiter des GNU-Projekts samt der daraus entstehenden, 1985 gegründeten Free Software Foundation (FSF). GNU sollte ein komplett freies und offenes Betriebssystem sein. Frei sollte es nicht im Sinne von „Freibier" sein, sondern im Sinne von „Freiheit". Es sollte eine Reihe von Anwendungen umfassen, die es jedem Benutzer ermöglichen würde, die Software uneingeschränkt zu nutzen, einzusehen, zu verändern und in veränderter Form weiterzugeben.[20] Aus diesem Wunsch heraus entstand die GNU-GPL, die GNU General Public License, mit der das Konzept einer „Freien Software" erstmals in eine Lizenz gegossen wurde.

Innerhalb des GNU-Projekts entstanden danach schnell zahlreiche Anwendungen. Die garantierten Freiheiten (re-)etablierten die (universitäre) Kooperation über den Austausch. Stallmans Idee funktionierte: Die Mitarbeit des Einzelnen im Kleinen konnte sich lohnen, weil das System als Ganzes dauerhaft frei zur Verfügung stehen würde.

Dem Anspruch, ein ganz freies, vollständiges unix(artiges) Betriebssystem zu erstellen, konnte das GNU-Projekt aber nur mit einem eigenen Kernel gerecht werden. Diese zentrale Komponente eines jeden Betriebssystems verbindet die Module und macht sie ablauffähig. Solange es zur Nutzung all der freien Tools des GNU-Projektes immer noch eines kommerziellen Unix-Kernels bedurfte, war das Ziel nicht erreicht. Linus Torvald war es schließlich, der diesen Kernel Anfang der 1990er Jahre programmierte, ihn Linux nannte und dessen Entwicklung bis heute maßgeblich beeinflusst. Er stellte seinen Linux-Kernel von Anfang an als freie Software zur Verfügung, indem er den Code unter den Bedingungen der GPL verteilte. Und auch da fruchtete die Idee von Richard Stallman: viele weitere Informatiker auf der ganzen Welt wurden angeregt, sich zu beteiligen. So entstand schließlich in der Kombination mit den bereits fertig gestellten Anwendungen des GNU-Projektes das, was als Ganzes heute als LINUX bekannt ist.[21]

2.4.2 Von „Free" zu „Open"

Den **Anstoß für die Entstehung der Open-Source-Bewegung** bildete **Ende der 1990er Jahre** die Offenlegung des Netscape Navigators. Netscape konnte sich damals mit dem Browser nicht gegen Microsoft und die Dominanz des Internet Explorers durchsetzen. Darum sollte die Netscape Codebasis freigegeben und der Browser von einer freien Community weiter gepflegt werden. Heute kennen wir das Ergebnis als „Firefox" sowie das damit verbundene Mozilla-Projekt.

Doch zur Zeit dieser Freigabe eines zuerst kommerziellen Produktes stellte sich das Attribut „freie Software" in der Kommunikation mit Unternehmen als schwierig heraus, wenn man auf dieser Basis neue kommerzielle Geschäftsideen verfolgen wollte. Die mit „freier Software" verbundene Philosophie und das mittlerweile gängige Verständnis von Code als Firmengeheimnis wirkten abschreckend. Um mit den Führungsetagen und Entscheidern in Konzernen besser ins Gespräch zu kommen, wurde nach einer alternativen Bezeichnung gesucht. Geschäftsfreundlicher sollte sie klingen und weniger ideologisch behaftet daher kommen. Der Vorschlag, dafür den Namen „Open-Source-Software" zu verwenden, soll von Christine Peterson (Foresight Institute) stammen. Basierend auf diesem Namen wurde schließlich **1998 die Open-Source- Initiative (OSI)** von Bruce Perens, Eric S. Raymond und Tim O'Reilly gegründet, die Ideen der freien Software und die Debian Free Software Guidelines zur Open-ource-Definition zusammengeführt.[22]

Vom Gebrauch her gibt es keinen Unterschied zwischen freier Software und Open-Source-Software: Alle Lizenzen freier Software sind auch als Open-Source- Lizenzen gelistet.[23] Umgekehrt werden die

[20] vgl. https://www.gnu.org/philosophy/free-sw.html
[21] Das gesamte Betriebssystem wird auch als GNU/Linux bezeichnet, um zu unterstreichen, dass ein Betriebssystem nicht nur aus seinem Kernel besteht und dass die Tools aus dem GNU-Projekt einen gleich wichtigen Anteil an dem funktionierenden Ganzen haben. (Vgl. GNU-Projekt: Warum Freie Software besser ist als Open-Source-Software; URL: https://www.gnu.org/philosophy/free-software-for-freedom.de.html).
[22] Open Source Definition (URL: http://opensource.org/osd)
[23] vgl. OSI: Licenses by Name (URL: http://opensource.org/licenses/alphabetical

vier konstitutiven Freiheiten freier Software – will sagen: sie nutzen, verstehen, weitergeben und verbessern zu dürfen[24] – vollständig von der Open-Source- Definition abgedeckt. Die Unterschiede manifestieren sich in den Botschaften der Begriffe: „Open-Source-Software“ unterstreicht den praktischen Nutzen und die Entwicklungsmethode; „freie Software“ betont die gesellschaftlichen Vorteile.[25] Pointiert ausgedrückt sei unfreie Software„[...] für die Open-Source-Bewegung […] eine suboptimale Lösung[;] für die Freie-Software-Bewegung [… aber] ein soziales Problem und freie Software (dessen) Lösung“.[26] Jedenfalls solle man, wenn man an freie Software denke, „[...] an *frei* wie in *Redefreiheit* denken, nicht wie in *Freibier*.“[27]

Gelegentlich wird versucht, den Namenskonflikt über hybride Begriffe wie FLOSS (Free / Libre Open-Source-Software) und FOSS (Free and Open-Source-Software) aufzufangen.[28] Das passt zu dem Statement, dass „die Freie-Software- und die Open-Source-Bewegung […] so etwas wie zwei politische Lager innerhalb der Freie-Software-Gemeinschaft (seien)“.[29]

[24] vgl. GNU-Projekt: Was ist Freie Software? (URL: https://www.gnu.org/philosophy/free-sw.html)
[25] vgl. Stallman, Richard: Warum Open Source das Ziel von Freie Software verfehlt (URL: https://www.gnu.org/philosophy/open-source-misses-the-point.html)
[26] GNU-Projekt: Warum Freie Software besser ist als Open-Source-Software (URL: https://www.gnu.org/philosophy/free-software-for-freedom.de.html)
[27] GNU-Projekt: Was ist Freie Software? (URL: https://www.gnu.org/philosophy/free-sw.html)
[28] vgl. Stallman, Richard: FLOSS und FOSS (URL: https://www.gnu.org/philosophy/floss-and-foss.html)
[29] Stallman, Richard: Warum Freie Software besser ist als Open-Source-Software (URL: https://www.gnu.org/philosophy/free-software-for-freedom.de.html)

3. Geschäftsmodelle

Open-Source-Software wird seit jeher von positiven und negativen Verallgemeinerungen begleitet. Wie so oft haben Verallgemeinerungen einen wahren Kern – so auch der gern hervorgehobene Vorteil „Das kostet nichts“ und seine Kehrseite „Damit lässt sich kein Geld machen!“ Beide Aussagen haben zwar einen wahren Kern, sind für sich genommen aber schlicht falsch. Wer mit Open-Source-Software Geld verdienen will, tut gut daran, sich die Feinheiten klar zu machen.

Wahr ist, dass für die Nutzung von Open-Source-Software keine Nutzungsgebühren verlangt werden, also keine Lizenzgebühren anfallen dürfen. Diese Bedingung ist in der Open-Source-Definition verankert. Deshalb ist jede Art von Open-Source-Software im direkten geldlichen Sinne lizenzkostenfrei. Und insofern 'kostet das tatsächlich nichts'. Konsequenterweise kann der Lizenzverkauf kein Geschäftsmodell für den Umgang mit Open Source bilden. Indirekt kostet der Erwerb der Nutzungsrechte von Open-Source-Software allerdings doch etwas – wenn auch wenig. Dies ergibt sich dadurch, dass das Wenige tatsächlich umgesetzt werden muss, was die Lizenzen umgesetzt sehen wollen.

Darüber hinaus aber – und das ist der entscheidende Punkt – ist jede Vergütung für jeden denkbaren Dienst für und mit Open-Source-Software legal. Ob die Vergütung in ihrer Höhe durchsetzbar ist, ist eine Frage der Wirtschaftlichkeit. Ob der Service nachgefragt wird, ist eine Frage des Marktes.

Tatsächlich hat die Notwendigkeit, mit Open-Source-Software bzw. mit freier Software Geld zu verdienen, von Beginn der Bewegung an eine maßgebliche Rolle gespielt: Nachdem Richard M. Stallman sich entschieden hatte, mit dem GNU-Projekt ein wirklich freies Unix zu programmieren, kündigte er auch seine Arbeit am MIT-Institut. Er wollte vermeiden, dass das Ergebnis seiner neuen Arbeit indirekt zum Eigentum des Instituts wurde, und dass die intendierten Freiheiten unterlaufen werden könnten. Konsequenterweise musste er seinen Lebensunterhalt auf eine neue Weise verdienen. Das tat er, indem er seinen Allround-Editor Emacs unter den von ihm selbst definierten Bedingungen freier Software auf Magnetbändern versendete. Geld verlangte er nicht für das Recht, den Editor Emacs zu nutzen. Mehr noch: auf den Bändern war sogar der Emacs-Quellcode. Bezahlen ließ er sich 'nur' für das Kopieren und den Versand der Bänder. Letztlich war Stallman selbst der erste kommerzielle Distributor freier Software.

Dieses erste Beispiel kann auch verdeutlichen, warum sich die Kosten für eine Distribution freier Software im Speziellen und für Open-Source-basierte Services im Allgemeinen immer auf eine spezielle Art und Weise „einpendeln“ werden:

Prinzipiell darf ein Distributor beliebige Summen für seinen Service verlangen. Die Lizenzen und die Open-Source-Idee sagen dazu nichts. Der Distributor muss nur einen Kunden für sein Produkt finden. Allerdings erhält dieser Kunde mit dem Kauf der Distribution nicht nur die Programme selbst, sondern auch die Freiheit, sie ganz oder teilweise an Dritte weiterzugeben. Diese Freiheit zur Weitergabe ist konstitutiv mit der Idee „Open-Source-Software' verbunden (vgl. Kapitel 2.2) Zudem liegt bei manchen der Programme deren Quellcode der Distribution direkt bei. Für die anderen könnte sich der Kunde den Quellcode auf denselben Wegen besorgen, derer sich sein Distributor bedient hat. Der Kunde bekommt also mit dem Kauf der Distribution alle rechtlichen und faktischen Möglichkeiten, selbst ein Distributor zu werden. Und diesen Dienst dürfte er sich dann auch selbst vergüten lassen. Im Falle von Stallman erhielten seine Kunden also immer auch das Recht, selbst Bänder mit Emacs zu vertreiben.

Generell muss ein Erbringer von Open-Source-Services bei der Preisbildung bedenken, dass sein Kunde (und jeder andere) sich die Open-Source-Software aus denselben Quellen besorgen kann, wie er selbst. Übersteigt die realisierbare Gewinnspanne für seinen Service die Kosten erheblich, die für den Aufbau des Services anfielen, werden Konkurrenten angelockt. Die Software selbst jedenfalls bietet im Open-Source-Umfeld keinen Schutz davor. Bei einem gut austarierten Preis- / Leistungs-Verhältnis wird die Kompetenz des Distributors zum Ausschlag gebenden Argument. Zu Zeiten der Emacs-Bänder hätte das bedeutet, dass die Kunden die Bänder anstatt beim Konkurrenten lieber

beim echten Kenner der Programms bestellt hätten, der immer die wirklich aktuellen Versionen hätte liefern können: also bei Stallmann.

Welches Geschäftsmodell man sich auch immer ausdenkt, ein Aspekt zeichnet den kommerziellen Umgang mit Open-Source-Software aus: Sie ist an sich nicht dazu geeignet, ein Alleinstellungsmerkmal aufzubauen. Ihre Zugänglichkeit und die mit ihr verknüpften Freiheitsrechte erleichtern das Entstehen von Alternativen. Jeder Open-Source-Geschäftsmann sollte die Software, auf der sein Service basiert, seinen Kunden im vollen Bewusstsein bereitstellen, dass er als Lieferant – bezogen auf die Software – leicht ersetzbar ist. Das ist allerdings kein Nachteil, sondern ein verkaufsfördernder Vorteil: Open-Source-Services bringen nicht nur Freiheit für die Softwarenutzer selbst, sondern geben dem Kunden auch einen größeren Grad an Freiheit im Verhältnis zu seinem Lieferanten. Umgekehrt wird der Open-Source-Geschäftsmann seinen Service im Bewusstsein seiner besonderen Kompetenz erbringen und daraus sein Alleinstellungsmerkmal formen. So gesehen war und ist Open-Source-Software immer schon ein besonderes marktwirtschaftliches Instrument.

Open-Source-Software ist also aus sich heraus kein Geschäftsmodell, sondern „nur“ eine besondere, kooperative Entwicklungsmethode. So begleitet die Entwicklung der Szene durchgängig die Frage, wie genau solche Geschäftsmodelle mit einer Ausrichtung auf Open Source aussehen können. Von Unternehmensseite wurde beispielsweise lange bezweifelt, ob für die „freie“ Software professioneller kommerzieller Support überhaupt möglich sei. Fraglich war zunächst auch, ob es wirtschaftlich überhaupt sinnvoll sein kann, OSS über die reine Nutzung hinaus auch in Form von Entwicklungsaufträgen oder über eigene Beteiligung zu unterstützen. Umgekehrt zweifelten OSS-Communities, ob sich Firmen an die Spielregeln der freien Software halten würden und ob eine kommerzielle Verwertung sich mit dem Geist der Gemeinschaften würde vereinbaren lassen.

Trotzdem war die Entstehung von Open-Source-Software immer schon mit kommerziellen Interessen verschränkt. Die 1989 gegründete Firma Cygnus Solutions (seit Ende 1999 Teil von Red Hat) gilt als eines der ersten Unternehmen, das vollständig auf ein Open-Source-Geschäftsmodell gesetzt hat. Selbst IBM hat mit der Integration des Apache-Webservers in seine Produktsuite einen frühen Beitrag zu dieser Verschränkung geleistet. Seit dieser Zeit ist ein diversifiziertes kommerzielles Open-Source-Ökosystem entstanden: gängige und akzeptierte Geschäftsmodelle haben sich herauskristallisiert; fortlaufend entstehen neue; andere – etwa die Distribution von neuen Releases auf Datenträgern – fokussieren sich des technischen Fortschritts wegen auf neue Bereiche. Auch haben sich Open-Source-Firmen in Verbänden organisiert: So existiert die Open Source Business Alliance (OSBA)[30] und auch innerhalb des BITKOM e.V. gibt es den etablierten Arbeitskreis Open Source.[31]

Die existierenden Geschäftsmodelle lassen sich in verschiedener Hinsicht klassifizieren: Zunächst ist die Unterscheidung in Services und Produkte möglich. Sodann lassen sich die Services typisieren: Auf der einen Seite stehen die Dienste, die mit Open-Source-Software erbracht werden. Auf der anderen stehen die, über die ein professioneller Betrieb von Open-Source-Software gewährleistet wird. Beide Formen der Dienste können bzgl. der Bezahlmodelle unterschieden werden. Und schließlich gibt es Produkte, in denen Open-Source-Software verwendet wird.

3.1 Services mit Open-Source-Software

Ein vertrautes Beispiel für Services, die mit Open-Source-Software erbracht werden, ist das Modell Web-Hoster: Firmen machen Webauftritte zugänglich und nutzen dafür Open-Source-Software. Kunden bezahlen dafür, dass ihr Webauftritt im Internet präsent ist. Ob das mit Open- oder Closed Source Software umgesetzt wird, ist für die eigentliche Leistung unerheblich. Open-Source-Software fließt als begünstigender Kostenfaktor in das Geschäftsmodell ein.

[30] vgl. http://www.osb-alliance.de/
[31] vgl. http://www.bitkom.org/de/wir_ueber_uns/65802.aspx

Dieses Konstrukt ist vielfach ausbaubar: Die Erstellung von Druckvorlagen in der Werbebranche, die Bildbearbeitung beim Fotografen, der Betrieb von Tonstudios, usw. Selbst der Betrieb von Rechenzentren, Netzzugängen und Cloudservices gehört in diese Kategorie.

3.2 Services für Open-Source-Software

Dem gegenüber stehen die Geschäftsmodelle, bei denen Services zum adäquaten Betrieb von Open-Source-Software vergütet werden. Die anbietenden Firmen müssen dabei nicht zwingend selbst Open-Source-Software entwickeln. Für Anbieter eigener Open-Source-Software (Applikationsanbieter) können weitere Serviceangebote ein sekundäres Gewinnmodell sein. Diese Services lassen sich wie folgt typisieren.

3.2.1 Support

Hierunter fällt sowohl die technische als auch die nicht-technische Kundenbetreuung, die in der Regel über einen Helpdesk oder ein Call-Center verwirklicht wird. Supportdienstleister können sich auf die technischen Aspekte der Installation und Integration neuer Software in bestehende Systeme spezialisieren, ebenso aber Unterstützung bei alltäglichen Anwendungsproblemen bieten.

Häufig werden Supportverträge abgeschlossen, die verschiedene Supportstufen, Verfügbarkeiten der Beratung und Reaktionszeiten festlegen. Supportstufen umfassen dabei First-, Second- und Third-Level-Support, jede Stufe stellt eine weitere Eskalation innerhalb des Systems dar. Eine Erreichbarkeit kann z.B. über Notrufnummern rund um die Uhr vereinbart werden, ebenso Reaktions- und Bearbeitungszeiten von z.B. maximal zwei Stunden. Festgeschrieben wird außerdem, ob eine Rechnungsstellung nach Abruf erfolgt oder aber ein zuvor bezahltes Kontingent eingelöst werden kann.

3.2.2 Entwicklung

Bei der kundenspezifischen Weiter- bzw. Auftragsentwicklung integriert der Dienstleister bestimmte Funktionen oder Gestaltungsmerkmale in die Open-Source-Software. So entstehen z.B. auf Basis eines bereits existierenden Softwarestandes eigenständige Anwendungen, die auf den Auftraggeber zugeschnitten sind. Ein solches Angebot für Open-Source-Software ermöglicht auch, unabhängig von einer z.B. Community-getriebenen Roadmap spezielle Feature (früher) zu verwirklichen, die für eine Leistung mit Open-Source-Software benötigt werden. Bezogen auf das Modell „Open-Source-Software“ besteht der Service dann u.U. nicht nur in der reinen Weiterentwicklung der Software, sondern auch darin, den Kontakt mit der Open-Source-Community, die Kooperation oder die Rückveröffentlichung von Entwicklungsständen zu organisieren.

3.2.3 Wartung

Wartung (Maintenance) von bestehenden Systemen ist meist festgelegt auf einen bestimmten Zeitraum (z.B. ein Jahr) sowie definierten Leistungsumfang (z.B. Einspielen regelmäßiger Sicherheitsupdates). Auch nicht mehr aktiv von Entwicklerseite unterstützte Software wird oftmals noch in komplexeren IT-Architekturen eingesetzt. Fällt der so genannte „Long Term Support“ offiziell weg, können Aufgaben von Drittanbietern übernommen werden, um eine Software oder Infrastruktur lauffähig zu halten.

3.2.4 Beratung und Schulung

Training (Schulung) und Consulting (Beratung) vervollständigen das Angebot um Open-Source-Software herum. Zahlreiche Anbieter – vor allem KMU oder Einzelpersonen – haben sich darauf spezialisiert, eine breite Palette von Kursen und Zertifizierungen anzubieten. Zielgruppe sind sowohl Anwender als auch Administratoren und Programmierer. Unternehmen können ihre Mitarbeiter so gezielt und professionell betreut weiterbilden sowie internes Know-how mehren. Auch große OSS-Anbieter zählen Schulungen zu ihrem Repertoire: Dadurch verbreiten und verbreitern sie z.B. das Wissen um ihr Open-Source-Angebot und bieten zeitgleich Service für Kunden.

Consulting-Angebote können Studien, Analysen und Konzeptionen umfassen. Sie zielen sowohl auf die Phase vor der Einführung einer Software (z.B. Auswahl) als auch auf deren Begleitung, um die Übergangszeit sowie den späteren Einsatz erfolgreich zu gestalten und Unternehmensprozesse anzupassen (z.B. Erstellung und Umsetzung eines Sicherheitskonzeptes). Ebenso finden sich Berater, die Firmen bei deren eigenen Software-Projekten betreuen, z.B. wenn erstmals OSS veröffentlicht werden soll.

3.3 Subskription

Prinzipiell kann jeder Dienst als Einzelleistung oder als eine wiederholt über einen bestimmten Zeitraum erbrachte Leistung vertrieben werden. Allerdings ist dem Subskriptionsmodell im Open-Source-Umfeld eine besondere Rolle zugewachsen: Bei Closed Software wird die Erlaubnis zur Nutzung der Software gesondert vergütet. Unternehmen, die für ihren Geschäftsbetrieb Closed Software einsetzen, binden sich in besonderer Weise an den Hersteller dieser Software. Eine solche Abhängigkeit entfällt bei der Verwendung von Open-Source-Software. In diesem Bereich besteht von der Sache her keine dauerhafte oder längerfristige Lieferantenbeziehung. Open-Source-Software kann weiter verwendet werden, auch wenn der Erbringer des in Anspruch genommenen Service wechselt. Deshalb muss sich der Open-Source-Dienstleister auf der anderen Seite in besonderer Weise auf die wiederkehrende Beauftragung konzentrieren. Kundenbindung zwischen Serviceerbringer und Kunden entsteht in einer Open-Source-basierten B2B-Kette durch die Qualität des Services, nicht allein durch die Qualität der Software.

Subskriptionsmodelle bieten die Bereitstellung von Software oder Services auf Zeit an. Subskriptionen können sich etwa auf die Bereitstellung von Softwareupdates oder den Zugriff auf Verzeichnisse (Repositories) beziehen. Auch die Bereitstellung von Handbüchern oder Anleitungen samt der passenden Aktualisierung können mit diesem Modell erfasst werden.

Vor allem der Mittelstand und damit die breite Masse von IT-Anbietern bedient sich des Subskriptionsmodells. Aber auch große Anbieter mit einem Enterprise Linux setzen darauf. Cloud-Lösungen sind vor allem bei großen Anbietern zu finden, die eine ausreichend skalierende Verteilung ihrer Softwareangebote über eine Cloudinfrastruktur sicherstellen können.

3.4 Duale Lizenzierung

Das duale Lizenzmodell bzw. die Mehrfachlizenzierung[32] bezieht sich auf die Weitergabe einer Software unter verschiedenen Lizenzen: Nutzer, die nicht den Pflichten der OSS-Lizenzen gerecht werden wollen, können die Software dann auch als Closed-Software gegen Entgelt erwerben.

Dies kann für beide Parteien sinnvoll sein: Der Softwarekunde kann, wenn ihm die Open-Source-Verpflichtungen nicht in sein Geschäftsmodell passen, Nutzungsrechte an einer Version gegen Entgelt

[32] vgl. dazu auch Ziffer 6.1.3 „Multiple Lizenzierung“: Grundsätzlich gibt es diese Art der Lizenzierung auch rein Open-Source-intern: Dann geht es den Entwicklern einer Software darum, über die mehrfache Lizenzierung unter verschiedenen Open-Source-Lizenzen eine besondere Sicherheit im Hinblick auf die Lizenzkompatibilität zu gewährleisten.

erwerben. So muss er die normalerweise mit der Software verbundenen Open-Source-bedingten Verpflichtungen nicht erfüllen. Und der Softwarehersteller kann spezielle Features, deren Programmierung er der seltenen Nachfrage wegen nicht über den Verkauf von Services abzudecken vermag, direkt den Nutzern in Rechnung stellen. Voraussetzung für eine Mehrfachlizenzierung ist natürlich, dass alle Urheber bzw. Urheberrechtsinhaber solch einer Mehrfachverwendung zustimmen bzw. vorsorglich zugestimmt haben.

Ein Spezialfall der Mehrfachlizenzierung dient aber der Etablierung eines besonderen Geschäftsmodells. Dabei wird die Grundversion einer Software unter einer OSI-zertifizierten Open-Source-Lizenz distribuiert. Die erweiterte Version mit „höherwertigen“, weil z.B. seltener nachgefragten Features, wird nach traditionellem Softwarelizenzmodell bereitgestellt.

Bei einem anderen Sonderfall dieses Ansatzes werden die Nutzungsrechte für die jeweils neueste Version – sozusagen vorab – gegen Entgelt eingeräumt, bevor sie nach einer Frist als Open-Source-Software frei zur Verfügung gestellt werden.

Bei der Duallizenzierung ist jedoch Vorsicht geboten, und zwar seitens des Lizenzgebers und des Lizenznehmers: Nicht alle Open-Source-Lizenzen erlauben es, dass nachträglich eine Duallizenzierung vorgenommen wird. Und manche „Grundversion“ ist tatsächlich so arm an Funktionalitäten, dass man kaum von funktionierender Software sprechen kann. Gleichwohl: wenn sie unter einer Open-Source-Lizenz freigegeben ist, ist sie Open-Source-Software. Der Kunde kann sich entscheiden, ob er sich dem Nachreichen der eigentlichen Funktionalität zu Bedingungen von Closed Software aussetzen will.

3.5 Verbundangebote

Gerade Open-Source-Software agiert nicht isoliert oder eindimensional. So kann eine bestimmte Konfiguration einer Software für eine Firma von besonderem Interesse sein, obwohl sie mit der Software selbst oder mit darauf aufbauenden Services von anderen Firmen allenfalls mittelbar etwas zu tun hat. Als eines der prominentesten Beispiele dafür gilt die Kooperation zwischen Mozilla bzw. Firefox und Google. 280 Millionen Euro soll Google 2012 an die Mozilla Foundation gezahlt haben, um als Standard für die Browsersuche gesetzt zu sein. Ende 2014 wechselte Mozilla zu einem nach Ländern aufgesplitteten Partnermodell, d.h. das unterschiedliche Suchmaschinen als Standards gesetzt sind.

Auch so genannte Mediatoren zählen in diese Kategorie. Sie bringen verschiedene Interessengruppen im Umfeld von Open-Source-Software über einen Marktplatz zusammen (Entwickler, Nutzer, Dienstleister, Werbetreibende). Dies praktiziert z.B. SourceForge, ein Tochterunternehmen des Appliance-Herstellers VA Software. SourceForge stellt eine technische Infrastruktur zur Verfügung, mit der Entwickler Software schreiben, ablegen, verwalten und kompilieren und die Entwickler miteinander kommunizieren können. Nutzer können über Projekt-Homepages Softwarepakete, Anleitungen und Dokumentationen herunterladen.

3.6 Weitere Modelle

Die vorstehende Auflistung kann nicht abschließend sein. Gerade der Open-Source-Gedanke ist es, der Geschäftsmodelle abseits klassischer Pfade eröffnet und der Kreativität Spielraum lässt. So zählt z.B. ein Linuxhotel bei Essen zu den innovativeren Geschäftsideen, auch wenn das dahinter stehende Geschäftsmodell prinzipiell dem Bereich Schulungen zuzuordnen ist.

Die zunehmende Verbreitung von OSS führt außerdem dazu, dass die Nachfrage nach Handbüchern, Dokumentationen und Anleitungen steigt. Davon profitiert beispielsweise ein Verlag wie O‘Reilly, der sich nicht nur schwerpunktmäßig mit Open-Source-Software beschäftigt, sondern auch von Bill O‘Reilly, einem bekannten Verfechter der OS-Bewegung, gegründet wurde.

Auch der Vertrieb von Merchandising wie dem Linux-Tux in Plüsch, als Schlüsselanhänger oder in diversen weiteren Ausführungen kann herangezogen werden. So setzen etwa die Mozilla Foundation und die Wikimedia Foundation auf den Verkauf von T-Shirts, Tassen, etc. mit entsprechenden Markenzeichen ihrer Projekte und Slogans. Damit kommt nicht nur ein wenig Geld in die Kasse: Zeitgleich kann auch die User-Community und die Nachfrage nach Fanartikeln bedient werden.

Schließlich gibt es noch das Spendenmodell: Als besonders nachhaltiger, finanzieller und werbewirksamer Erfolg darf das Projekt der Mozilla Foundation zum Start des Firefox 1.0 gewertet werden: 2004 initiierte die Stiftung eine Kampagne, um den alternativen Browser bekannt zu machen. Mit dem bei Unterstützerinnen und Unterstützern eingesammelten Geld konnten einseitige Anzeigen für den Browser in auflagenstarken Tageszeitungen geschaltet werden. Zehn Jahre später wird auf dem Schweizer Open-Source-Business-Forum eine Software ausgezeichnet, dessen Entwickler auch das Geschäftsmodell der Donation verwendet.[33]

[33] Vgl. CH Open Source Awards: Open Source Business Forum und Preisverleihung OSS Awards 2014 (URL:http://www.ossawards.ch/preisverleihung2014/)

4. Erstellungsmodelle[34]

So, wie die Frage nach Open-Source-Geschäftsmodellen manchmal mit irrigen Verallgemeinerungen beantwortet wird, wird gelegentlich auch die Frage nach dem Entstehen von Open-Source-Software klischeehaft falsch beantwortet. Es ist eben nicht richtig, dass Open-Source-Software nur von Idealisten programmiert wird, die ihre Freizeit dafür opfern – auch wenn es sicher viele Idealisten unter den Open-Source-Programmierern gibt und auch wenn diese oft und viel von ihrer Freizeit für diese Art der Arbeit „opfern".

Wie schon mehrfach angeführt, steht der Begriff Open-Source-Software eben nicht im Konflikt zu kommerzieller Software: OSS kann auch kommerziell entwickelt, kommerziell vertrieben und kommerziell eingesetzt werden. Im Gegensatz zu Closed Software wird Open-Source-Software aber auch nicht-kommerziell entwickelt. Mehr noch: gelegentlich überlagern sich die private und die kommerzielle Weiterentwicklung; Firmen und Privatentwickler arbeiten dann zeitgleich an demselben Produkt. Trotz unterschiedlicher Interessen sind alle Beteiligten dem Freiwilligkeitsprinzip unterworfen. Die Kooperation ergibt sich, sie ist nicht autoritativ gelenkt. Führung wird von den Geführten gewährt. Sie erwächst aus attestierter Kompetenz. Und sie ist volatil. Das zeichnet eine Open-Source-Community aus. So müssen die organisatorischen und sozialen Prozesse anders gestaltet sein als bei einer rein firmengelenkten Entwicklung. Und firmengelenkte Entwicklungen werden sich da, wo sie mit der Open-Source-Community interagieren, an die anderen Gegebenheiten anpassen müssen.

Möge der Einfachheit halber das Wort „Business" für eine Unternehmensstruktur mit zentraler Leitung und Zielvorgaben stehen samt einer Softwareentwicklung, die mit den Parametern Kosten, Ressourcen und Terminen gesteuert wird. Dann darf unter einer „Community" eine Kooperation von Individuen und Unternehmen verstanden werden, die auf Freiwilligkeit und Teamkonsens basiert. Dabei sind einzelne Beiträge nicht einforder- und kaum planbar, sondern werden von den Teilnehmern nach eigenem Ermessen und Zeitbudget erbracht. Entsprechend sind die Zielparameter für Communities nur Ressourcen und Zeit.

Auffällig ist nun, dass die reine Lizenzierung als Open-Source-Software weder das eine noch das andere Modell präferiert. Auch wenn, historisch gesehen, die Community-getriebene Entwicklung im Vordergrund stand und steht, kann eine Open-Source-Software sehr wohl auch firmengetrieben entwickelt werden (vgl. zur Entwicklung von OSS in Firmen auch Ziffer 7.2.1 „Modelle der Erstellung und Bereitstellung"). In diesem Sinne kann wie folgt unterschieden werden:

a) Community-driven development (CDD): Die Entwicklung liegt in den Händen der Gemeinschaft. Sie wird von einer von Einzelinteressen unabhängigen und oft non-profit-getriebenen Core Group betreut. Diese setzt sich meist aus Personen zusammen, die ihr Engagement und ihr Expertentum bereits unter Beweis gestellt haben. Die Core Group steuert die Veröffentlichung neuer Versionen (Releases) und trifft Richtungsentscheidungen. Entwicklungen orientieren sich am Bedarf und den Wünschen, die in der Gemeinschaft festgestellt und priorisiert werden. Gemeinschaft kann hierbei im Sinne eines Multi-Stakeholder-Ansatzes sehr weit gefasst sein und auch Firmen als Akteure umfassen.

b) Business-driven development (BDD): Die Entwicklung der Software ist absatzorientiert ausgerichtet, orientiert sich am Markt und den Anforderungen, die von Kunden gestellt werden. Entwickler sind in der Regel bei Firmen angestellt. Eine Community bildet sich u.U. aus den firmeneigenen und externen Entwicklern und Anwendern. Der Entwicklungsprozess wird in der Regel transparent gehalten (öffentliche Roadmap, Quellcode und Bugtracking, einsehbares Ticketsystem, etc.) und ist offen für externe Quellcodebeiträge, oft auf der Grundlage von mit Contributor-Agreements (vgl. Ziffer 6.12.1) erbracht werden.

Beide Modelle haben ihre Vor- und Nachteile, die im Folgenden dargestellt werden.

---

[34] Dieses Kapitel behandelt rein die grundsätzliche Unterscheidung zwischen firmen- und Community-gelenkter Entwicklung. Welche Möglichkeiten zur konkreten Ausgestaltung des Erstellungsmodells Firmen in Zusammenhang mit OSS haben, erläutert Ziffer 7.2.1 „Modelle der Erstellung und Bereitstellung" im Detail

4.1 Open-Source-Software aus Community-gestützter Entwicklung

Stärken der Community-getriebenen Entwicklung liegen etwa in den Punkten:

- Skalierung: Die Zahl potenziell zu einem Projekt Beitragender ist hoch. Dadurch kann auf viel Know-how zugegriffen werden; in der Summe sind mehr eingesetzte Stunden und mehr eingebrachte Codezeilen möglich. Dank der Kommunikationsmöglichkeiten über das Internet sind schnelle Verständigung und testgetriebene Entwicklung selbstverständlich geworden.
- Redundanz: Know-how zu bestimmten Fragestellungen ist häufig bei mehreren Personen vorhanden, die sich gegenseitig unterstützen und einspringen können, wenn schneller Einsatz gefragt ist. Für ein Problem können so parallel auch mehrere Lösungen entstehen, idealerweise setzt sich die von der Community bevorzugte durch.
- Delphi-Effekt: Die Zuverlässigkeit von Expertenentscheidungen steigt mit Zahl der Beteiligten.
- Preis-Leistungs-Verhältnis: Dieser Faktor ist unschlagbar, da viele ihre Zeit und ihre Kenntnisse auf freiwilliger Basis und unentgeltlich zu Verfügung stellen.

Einige Stärken der Community können sich je nach Situation allerdings auch in Schwächen verwandeln. Letztlich verursacht werden sie durch den Aspekt der Freiwilligkeit:

- Einzelfallzentrierung: Unter dem Stichwort „works for me“ kann es zu einer Fokussierung auf die eigenen Probleme und deren Lösung kommen. Was andere benötigen, tritt in den Hintergrund. Möglicherweise entstehen so individuell tragbare Lösungen, die aber für andere nur schwerlich anwendbar sind.
- „Not invented here“: Mit der Konzentration auf die eigene Befindlichkeit und die eigene Programmierlust besteht ohne eine äußere Korrekturinstanz die Gefahr, dass von außen an den etablierten Zirkel heran getragene Ideen und Lösungen ungerechtfertigt ignoriert werden.
- Koordinationsnotwendigkeit: Abseits von Firmenhierarchien wird die Koordination des Entwicklerteams nicht nur notwendiger, sondern auch komplizierter.

4.2 Open-Source-Software aus firmengesteuerter Entwicklung

Zu den Stärken einer firmengetriebenen Entwicklung gehören etwa:

- Gelenkte Prozesse: Durch einen Top-Down-Ansatz kann die Entwicklung von Open-Source-Software gezielter voran schreiten. Sie wird nicht allein durch Richtungsentscheidungen einer Core Group gelenkt, sondern auch von unternehmerischen Überlegungen geleitet.
- Konsortien: Mehrere Stakeholder können sich zusammenschließen, ihre Interessen koordinieren, Ressourcen koppeln und so ebenfalls zu einem fokussierten Vorgehen beitragen.
- Anwender-Fokus: Die Entwicklung wird nicht von den Interessen und Vorlieben des Einzelnen geprägt. Stattdessen stehen die Wünsche und Vorteile der Nutzer im Fokus, gerade auch, wenn es um User Experience, Grafik und Design geht.

Dennoch kann die Ansiedlung von OSS-Projekten im Firmenkontext auch Schwächen mit sich bringen:

- Skalierung: Die eingesetzten Ressourcen sind im Vergleich zur Community zwar gezielter einsetz- und steuerbar; aber sie sind auch deutlich knapper. Statt verteilter Ressourcen muss auf Mitarbeiter zugegriffen werden, die in Firmenprozesse und -notwendigkeiten eingebunden sind.
- Preis-Leistung: Entwickler stehen bei der Firma / einer der Firmen unter Vertrag. Die Kosten unterscheiden sich also nicht von denen für jene Software, die für eigene Zwecke entwickelt wird. Damit unterliegen sie traditionellen Kalkulationsmechanismen und Steuerungen: Auftragsentwicklung, Bugfixing etc. muss in entsprechenden Stundensätzen berechnet und gerechtfertigt werden.
- Eindimensionalität: Kennzeichen erfolgreicher Community-Entwicklungen ist oft der Mut zu Irrwegen. In Firmenkontexten ist die Fehlerkultur hingegen eine andere.

5. Standardisierung und Kundenschutz

5.1 Zertifizierungen

Zertifizierungen spielen im Bereich der Open-Source-Software eine nicht unerhebliche Rolle und leisten einen Beitrag zur Standardisierung. Zum einen können dabei die Zertifizierungen für (juristische) Personen betrachtet werden, zum anderen die für die zum Einsatz kommenden Open-Source-Lizenzen. Aus der Analyse möglicher Open-Source-Geschäftsmodelle (vgl. oben Kapitel 3) ergibt sich, dass Unternehmen für ihre auf OSS aufbauenden Angebote andere Kriterien benötigen, über die sie sich auszeichnen, als dies bei Closed-Source-Ansätzen der Fall ist, die auf Lizenzentgelten basieren. Ausgewiesene Expertise ist ein solches Kriterium, das Vorteile gegenüber Mitbewerbern schaffen kann. Die Qualifizierung der Mitarbeiter wird durch die abgelegten Prüfungen am Linux Professional Institut (LPI) deutlich.[35] In zugehörigen Kursen (teils auch im Selbststudium) eignen sie sich Wissen verschiedener Erfahrungsstufen an und erhalten im Erfolgsfall ein Zertifikat. Mit dem LPI, den Lernmaterialien und den entsprechenden Prüfungen existiert also eine einheitliche, standardisierte Wissensbasis. Das erleichtert auch das Verfassen von Stellenangeboten sowie die Einschätzung des Kenntnisstandes eines Bewerbers. Bei Unternehmen steht z.B. das Siegel „Red Hat Certified“, das diese erreichen und mit dem sie werben können, für besondere Kompetenzen im Bereich Red Hat Enterprise Linux (RHEL).

Der Begriff „Open Source“ selbst ist auch als Label erfolgreich und dient häufig als Marketinginstrument. Doch nicht alles, was als „Open Source“ deklariert ist, verdient diesen Titel auch – zumindest nicht nach OSD oder der Definition der FSF für freie Software. Die Zertifizierung von Lizenzen durch die OSI (vgl. Ziffer 2.2) erfüllt dabei eine wichtige Standardisierungsfunktion: Die Verwendung einer entsprechenden Lizenz garantiert Anbietern wie Nutzern von OSS Verlässlichkeit.

Gelegentlich taucht jedoch Open-Source-Software auf, für die auf Grundlage einer etablierten Lizenz aus der BSD-Lizenz-Familie eine eigene Lizenz erstellt wurde – diese kann unter Umständen aber selbst nicht mehr der OSD entsprechen. Die betroffene Software kann nach wie vor quelloffen sein, dafür aber mit anderen Einschränkungen verbunden sein wie z.B. das Verbot der kommerziellen Weiternutzung oder den Ausschluss bestimmter Nutzergruppen (z.B. Rüstungsfirmen, Geheimdienste, …). Anbieter sollten darum auf eine von der OSI zertifizierte Lizenz setzen – zumal sich unter den offiziell gelisteten für fast jeden Anwendungsfall eine entsprechende Lizenz findet. Nutzer sollten darauf achten, dass die Software unter einer ebensolchen Lizenz steht. Für beide Seiten minimiert dies Unsicherheiten und potenzielle Risiken. Andernfalls müssten ungewöhnliche Lizenzpflichten von nicht der OSD entsprechenden Lizenzen dauerhaft vom Nutzer erfüllt werden und ggf. auch mit der Software weitergegeben werden.

Mittlerweile sind Presse und Öffentlichkeit für Missbräuche des Labels Open-Source-Software sensibilisiert. Wer sich der Bezeichnungen „Open Source“ oder „freie Software“ bedient, sollte sich bewusst sein, dass Internet und soziale Medien schnell die Fälle aufdecken, in denen dies nur Fassade ist. Der potenzielle Image-Schaden kann höher sein als die Vorteile, die durch die Deklarierung erzielt werden sollten.

5.2 Supportleistungen für Open-Source-Software

Open-Source-Software unterscheidet sich in puncto Supportleistungen in der Regel von herkömmlicher Closed Software. Hauptgründe dafür sind zum einen eine nicht selten dezentrale Entwicklung der Software ohne konkreten Ansprechpartner für die Nutzer, zum anderen die meist kostenlose Überlassung der Software zur Nutzung oder Weiterentwicklung.

Während sich die Gewährleistung bei entgeltlich vertriebener Software nach den gesetzlichen Bestimmungen des Kauf- oder Werkvertragsrechts richtet, ist die Gewährleistung bei kostenlos überlassener Software wie Open-Source-Software eingeschränkt. Sie richtet sich im Regelfall nach

[35] vgl. https://www.lpi.org

den gesetzlichen Regelungen der Schenkung. Das hat zur Folge, dass der Hersteller für Mängel nur einstehen muss, wenn ihm Vorsatz oder grobe Fahrlässigkeit oder das arglistige Verschweigen eines Sach- oder Rechtsmangels vorzuwerfen ist. Ein Rechtsmangel liegt insbesondere vor, wenn der Nutzung der Software Rechte Dritter entgegenstehen. Dem Vorteil fehlender Lizenzkosten stehen daher eingeschränkte Gewährleistungsansprüche gegenüber.

Um die Lücke fehlender Gewährleistungsansprüche zu schließen, die bei entgeltfreier Software entsteht, kann es sich anbieten, entgeltliche Service- und Supportleistungen in Anspruch zu nehmen. Die Palette solcher Angebote ist vielfältig – sie reicht von der Fehlerbehebung über das Schließen von Sicherheitslücken bis hin zur Weiterentwicklung und individuellen Anpassung der Software.[36]

Je nach Inhalt der vertraglichen Vereinbarung kommen verschiedene Vertragstypen im Hinblick auf Service- und Supportleistungen in Betracht: Sind die Leistungen nach der Vereinbarung der Parteien erfolgsbezogen ausgestaltet, wird es sich regelmäßig um einen Werkvertrag handeln. In diesem Fall wird ein konkretes Leistungsergebnis bzw. ein Tätigkeitserfolg geschuldet, so beispielsweise bei der Beseitigung von Störungen bzw. Fehlern und der Aufrechterhaltung der Funktionsfähigkeit der OSS. Geht es hingegen lediglich um die gewissenhafte Durchführung der Maßnahmen, beispielsweise um schlichte Beratungsleistungen beim Einsatz von OSS, kommt ein Dienstvertrag in Betracht. Da bei diesem gerade kein fassbares Ergebnis geschuldet wird, sollte darauf geachtet werden, festzuschreiben, in welcher Qualität die Service- und Supportleistung zu erbringen ist.

Eine klare Einordnung als Werk- oder Dienstvertrag fällt nicht immer leicht und hängt letztlich vom Willen der Beteiligten ab. Die Beteiligten sollten darauf achten, die Vereinbarungen so eindeutig wie möglich zu gestalten, um auf diese Weise eine sichere Grundlage für die gemeinsame Zusammenarbeit zu schaffen und Unstimmigkeiten im Falle von Leistungsmängeln zu vermeiden. Ist ein Werkvertrag gewollt, so sollte insbesondere eine Abnahme ausdrücklich geregelt werden.

5.3 Schutz vor Ansprüchen Dritter

Beim Einsatz von OSS kann es zur Verletzung von Rechten Dritter kommen. Zu nennen sind hier insbesondere fremde Urheber- und Patentrechte, aber auch Markenrechte. Da OSS im Hinblick auf solche Rechtspositionen vor ihrem Einsatz in der Praxis meist nicht näher überprüft und analysiert wird, besteht das nicht zu unterschätzende Risiko von Rechtsverletzungen und damit von teuren Abmahnungen, Unterlassungs- und Schadensersatzklagen sowie kostenintensiven Patentrechtsstreitigkeiten, initiiert durch die jeweiligen Rechteinhaber. Treffen kann es letztlich jeden, den, der OSS vertreibt und den, der OSS nutzt.

Wie schon unter Ziffer 5.2 ausgeführt, haftet der Lizenzgeber im Fall kostenlos überlassener Software für Rechtsmängel nur eingeschränkt. Wurde weder vorsätzlich noch arglistig gehandelt – was oft der Fall sein dürfte –, bleibt der Anwender bzw. Distributor auf dem Schaden sitzen; in diesem Fall trägt er allein das Risiko von Mängeln.

Für diesen Fall bieten einige Distributoren einen speziellen, kostenpflichtigen Versicherungsschutz an. Solche Produkte, etwa unter dem Namen „Indemnification Program“ oder „Assurance Program“ vertrieben, räumen den OSS-Anwendern zusätzliche Ansprüche ein und sichern sie gegen die genannten Risiken finanziell ab. Solche Versicherungen decken in der Regel Schäden, die aufgrund von Patent- und Urheberrechtsverletzungen entstehen, in gewissem Umfang ab. In diesen Konstellationen sinkt das Risiko der Verletzung von Rechten Dritter auch dadurch, dass typischerweise mehr Aufwand beim Distributor in die Prüfung auf solche möglichen Rechte Dritter gesteckt wird.

[36] vgl. Kapitel 3 zu Geschäftsmodellen

Im Einzelfall sollte jedoch geprüft werden, ob diese Versicherungsprodukte auch die konkret beabsichtigte Nutzung der Software abdecken. So ist darauf zu achten, dass sich Ansprüche nicht lediglich auf den Fall der Eigennutzung beschränken, wenn ein Vertrieb der Software einzeln oder im Paket mit anderen Komponenten beabsichtigt ist. Auch gibt es den Fall, dass nur ein Kern der Distribution von der Versicherung erfasst ist, nicht jedoch sämtliche, in der Distribution enthaltenen Programmteile.

5.4 Long Term Support

Der ‚Long Term Support' bietet Anwendern eine besondere Art der Sicherheit: die Produktversionen werden längerfristig mit wichtigen Bugfixes und Security-Upgrades versorgt. Dass das eine besondere Herausforderung werden kann, hat folgende Gründe:

Open Source Software wird von der Community in doppelter Hinsicht kontinuierlich weiterentwickelt. Zum einen geht es um die Bereitstellung neuer funktional angereicherter Versionen der Software. Zum anderen geht es um die Bereitstellung von Bugfixes und Securityupgrades der funktional nicht erweiterten Version. Anwender, die nicht immer gleich auf die neuesten funktionalen Versionen umstellen wollen, müssen also ein Interesse haben, dass die von ihnen verwendete Version über einen längeren Zeitraum mit Bugfixes und Securityupgrades versehen wird, ohne dass damit ein funktionales Upgrade einherginge. Distributoren von Open Source-Software haben dafür das Konzept von ‚Long-Time-Support-Versionen' entwickelt. Hier sichert der Distributor für eine bestimmte Sammlung von OSS zu, dass er über einen längeren Zeitraum als üblich die erwünschten Bugfixes und Securityupgrades ausliefert. Oder anders gesagt: LTS-Versionen rücken zum Zweck der Risikominimierung die Softwarequalität in den Mittelpunkt, neue Features stehen zurück.

Stabile LTS-Versionen zeichnen sich in der Regel durch einen Feature-Freeze (dt.: Einfrieren der Funktionen) aus. D.h. der Status Quo wird beibehalten, während in den Source Code (und evtl. bereits neuere Versionen) neue Funktionen einfließen; Bugs und Schwachstellen werden nach wie vor behoben. Die entsprechenden Patches können einzeln oder in neuen Releases (Maintenance oder Service Packs, Minor Releases etc.) zur Verfügung gestellt werden. Die LTS-Versionen zeichnen sich dementsprechend in Art und Frequenz der Updates aus: So sind Updates seltener und bestehen – wenn überhaupt – aus Funktionen, die schon ausgiebig getestet wurden. Das soll die Gefahr minimieren, dass sich neue Bugs einschleichen oder bisherige Funktionen unabsichtlich beeinträchtigt bzw. sogar außer Betrieb gesetzt werden. Ein typischer Zyklus für eine solche LTS-Version liegt bei zwei Jahren – kann aber auch kürzer oder länger sein oder relativ zu weiteren Releases ausfallen.

Nur am Rand sei darauf hingewiesen, dass dies umgekehrt in dem Sinne auch Konsequenzen für Anbieter von OSS haben kann, als sie über ihre eigene LTS-Strategie nachdenken sollten. Diese umfasst sowohl die Dauer, für die alte Versionen noch unterstützt werden, als auch die Veröffentlichung von speziellen LTS-Versionen. Eine weiteres Geschäftsmodell besteht mittlerweile darin, kommerziellen LTS über den durch die Community gewährleisteten hinaus anzubieten.

Nutzer von OSS sollten sich über die jeweiligen Regeln des LTS informieren. Fast alle Projekte haben eine eigene Policy, wie etwa die Eclipse Foundation oder die von der Firma ‚Canonical' erstellte Distribution *Ubuntu*. Für die langfristige IT- und Produktplanung sind die daraus zu gewinnenden Informationen von hohem Wert.[37]

[37] Einen besonderen Fall stellen Hersteller von Customer Electronics dar, die Linux auf ihren Systemen zum Einsatz bringen: Hier wurde einer firmenübergreifende Long Term Support Initiative (LTSI) in Form eines ein industrieweites Projekt ins Leben gerufen. Dieses Projekt legt Long Term Support für einen Industrie-Zweig des Linux-Kernels fest, der als verlässliche Basis für Customer Electronics dient. (vgl. dazu etwa: https://ltsi.linuxfoundation.org/what-is-ltsi)

6. Rechtliche Aspekte von Open-Source-Software

6.1 Gang der Darstellung

Im folgenden Kapitel 6 werden die rechtlichen Grundlagen von Open-Source-Software und ihrer Nutzung dargestellt. Dabei erfolgt zunächst unter 6.2 und 6.3 eine Auseinandersetzung mit den wesentlichen Vorgaben der Open-Source-Lizenzen selbst. Diese werden kurz dargestellt und in den Lizenzkontext eingeordnet. Welche Implikationen die gesetzlichen Grundlagen in Deutschland insbesondere im Urheber- und Patentrecht für Open-Source-Software haben, lässt sich den Ausführungen unter 6.4 und 6.5 entnehmen. Im Anschluss daran wird der Rechtsrahmen für Software-Verträge einschließlich der Wechselbeziehungen mit den Vorgaben der Open-Source-Lizenzen dargestellt (6.6 bis 6.9). Dabei nimmt die Darstellung von Rechten und Pflichten aus OSS-Lizenzen einen breiten Raum ein. Es folgt die Betrachtung besonderer rechtlicher Fragestellungen zur Erstellung von Open-Source-Software unter Berücksichtigung arbeitsrechtlicher Aspekte (6.10) und zu den Anforderungen an die Ausschreibung von Open-Source-Software im Recht der öffentlichen Auftragsvergabe (6.11). Ein Überblick über die bisher ergangene Rechtsprechung zu Open-Source-Software findet sich in 6.12. Das Kapitel schließt mit einer Zusammenfassung der rechtlichen Aspekte (6.13) und Handlungsempfehlungen (6.14).

6.2 Lizenztypen

Ein Computerprogramm (Software) ist ein durch das Urheberrecht geschütztes Werk, an dem der Autor mit dem Akt der Erstellung umfangreiche Rechte erhält. Das deutsche Urhebergesetz (UrhG) behält dem Software-Ersteller weitgehend die Nutzungsmöglichkeiten seiner Software vor und verlangt zu seinem Schutz, dass die Nutzung der Software durch Dritte seine Zustimmung erfordert (§ 69c UrhG). So darf der Nutzer der Software diese grundsätzlich nur zur Anwendung bringen, wenn ihm die Nutzung im Einzelnen vertraglich vom Software-Ersteller eingeräumt wurde. Unabhängig von der vertraglichen Rechtseinräumung sind einige Nutzungsmöglichkeiten bereits von Gesetzes wegen gestattet. So ist dem Nutzer z.B. die Erstellung einer Kopie zum Zwecke der Sicherung (§ 69d Abs. 2 UrhG) und die Dekompilierung der Software zur Schnittstellenanpassung (§ 69e UrhG) erlaubt. Im Rahmen der bestimmungsgemäßen Nutzung einer Software darf der Nutzer auch eine Fehleranalyse und -behebung vornehmen, ohne dass er hierfür eine gesonderte Zustimmung des Software-Erstellers benötigt (§ 69d Abs. 1 UrhG).

Dem Nutzer von Open-Source-Software (OSS) werden von vornherein durch die Urheber weitergehende Rechte eingeräumt: das Recht auf beliebige Ausführung, das Recht auf Veränderung und das Recht auf Weiterverteilung der unveränderten oder veränderten Software. Die Non-Profit-Organisation „Open-Source- Initiative“ (OSI) hat den Begriff „Open-Source-Software“ eingeführt und in der „Open-Source- Definition“ festgelegt. Inhaltlich wurde dabei auf dem Begriff „Free Software“ der „Free Software Foundation“ (FSF) aufgesetzt. Im Folgenden wird ausschließlich der Begriff Open-Source-Software (OSS) verwendet.

Die Gewährung der Rechte an einer Open-Source-Software erfolgt über Lizenzen. Die beiden Software-Organisationen OSI und FSF bieten Zertifizierungsverfahren für Open-Source-Lizenzen an, um eine Lizenz als OSI-konform beziehungsweise FSF-konform zu bestätigen. Den vom Software-Urheber gewährten Rechten stehen auch bei Open-Source-Software immer Pflichten gegenüber, die der Nutzer bei Ausübung seiner Rechte erfüllen muss, um sich das Nutzungsrecht an der Software zu erhalten.

In diesem Zusammenhang sei darauf hingewiesen, dass insbesondere der Pool der von der OSI zertifizierten OSS-Lizenzen für fast jeden denkbaren Anwendungsfall bereits eine passende Lizenz enthält. Software-Ersteller müssen also keine weiteren OSS-Lizenzen kreieren. Bereits jetzt fällt es in der Praxis nicht leicht, über die verschiedenen Lizenzmodelle den Überblick zu behalten.

6.2.1 Klassifizierung an Hand des „Copyleft-Effekts“

Open-Source-Lizenzen werden in zwei große Klassen eingeteilt, die permissiven Lizenzen und die sogenannten Copyleft-Lizenzen.

„Copyleft“ bezeichnet die Pflicht, Änderungen und je nach Lizenz auch Ableitungen oder Einbettungen einer Software wieder unter dieselbe Lizenz wie die Ursprungssoftware zu stellen. Die gesamte Software muss also einschließlich aller Änderungen und Ergänzungen wiederum nach den Vorgaben der Lizenz für die Ursprungssoftware zur Nutzung freigegeben werden. Im Falle einer Open-Source-Lizenz bedeutet dies, dass der Quellcode von Änderungen und – je nach Lizenz – auch von Ableitungen oder Einbettungen offen zu legen ist. Die Offenlegungspflicht kann auch proprietären Quellcode betreffen, also Code, der ursprünglich nicht als Open-Source-Software gedacht war. Diese Wirkung der Open-Source-Lizenz wird in der Praxis auch teilweise undifferenziert als „viraler“ Effekt bezeichnet.

Die Reichweite des Copylefts kann stark oder schwach sein. Entsprechend bürgerte sich die Unterklassifizierung in starke und schwache Copyleft-Lizenzen ein. Im Falle des schwachen Copylefts findet sich in den Lizenzen ein - im Detail oft unterschiedliches - Abgrenzungskriterium, das den Copyleft-Effekt in der einen oder anderen Form auf die Änderungen an der OSS selbst begrenzt. Das bedeutet, die tatsächliche Reichweite des Copyleft-Effekts muss jeweils im konkreten Fall ermittelt werden. Anhaltspunkte dafür sind der Lizenztext, das jeweils anzuwendende Urheberrecht, die Absichten von Lizenzverfasser und Lizenzgeber (soweit für den Adressaten erkennbar) sowie die technische Einsatzweise im Zusammenspiel mit anderen Softwarebestandteilen.

In die Gruppe der Lizenzen mit starkem Copyleft-Effekt gehört die „Mutter“ aller Copyleft-Lizenzen, die „GNU General Public License“, sowie die „GNU Affero General Public License“ und die „CeCill“-Lizenz.

Prominente Beispiele für Lizenzen mit schwachem Copyleft-Effekt sind die „Lesser GNU Public License“, die „Mozilla Public License“ und „Common Development and Distribution License“. Die Lizenzen „Eclipse Public Licence“, „European Public Licence“ und „Common Public License“ werden je nach Rechtsraum unterschiedlich als mit starkem oder schwachem Copyleft-Effekt behaftet gesehen.

Im Gegensatz zu den Copyleft-Lizenzen erfordern die permissiven Lizenzen keine Bereitstellung des Quellcodes. Prominente Beispiele für permissive Lizenzen sind die „Berkeley Distribution License“, die MIT-Lizenz des Massachusetts Institute of Technology, die „Apache License“ und die „Microsoft Public License“.

Die Klassifizierung von Lizenzen bzw. von Software kann zwar die Orientierung erleichtern. Für die korrekte Rechtsanwendung können aus einer solchen Klassifizierung jedoch keine verbindlichen Anhaltspunkte abgeleitet werden, da es dafür maßgeblich auf den Inhalt der Lizenzbedingungen und ihre Interpretation ankommt. Eine ausführliche Darstellung von Entstehung und Wirkungsweise des intuitiv schwer fassbaren Konzepts des Copyleft-Effekts enthält Kapitel 2.3.

6.2.2 Subklassifikation an Hand von Patentklauseln

Eine weitere Grobklassifizierung der Lizenztypen lässt sich im Zusammenhang mit Patentklauseln vornehmen. Die Antwort auf die Frage, warum Patentklauseln in modernen Open-Source-Lizenzen verbreitet sind, mag sich nicht auf den ersten Blick erschließen. Hierzu muss man wissen, dass die meisten Open-Source-Lizenzen vor dem Hintergrund des US-amerikanischen Rechts entwickelt wurden. Leichter als im europäischen Rechtsraum kann in den USA Software nicht nur mit dem Instrument des Urheberrechts, sondern zusätzlich mit dem Instrument des Patentrechts geschützt werden. Damit dieser „zweigleisige“ Schutz nicht dazu führt, dass das mittels urheberrechtlicher Regelungen erreichte Schutzniveau über das Patentwesen ausgehebelt wird, sehen mittlerweile eine Vielzahl von Open-Source-Lizenzen ausdrückliche Patentklauseln vor. Diese lassen sich in die folgenden Kategorien unterteilen:

Kommentiert [KT1]: Kommentar Frau Schnizer: Aus den Kommentaren von Hrn. Teichert und der Diskussion am 2.9. habe ich gesehen, dass es in dieser Passage wirklich Änderungsbedarf gibt.

In die Änderungen sind folgende Überlegungen eingegangen:
- Das Abgrenzungskriterium für den schwachen Copyleft-Effekt sollte erwähnt werden, auch wenn es nicht immer gleich ausgeprägt ist. Dieses Vorgehen bringt auch die Darstellungsweise in Kapitel 2.3 und 6.1.1 näher zueinander.
- In der Beschreibung wird bereits klar gemacht, dass Ableitungen/Einbindungen von der Offenlegungspflicht betroffen sein können, aber nicht zwingend sein müssen.
- Expliziter Verweis auf die ausführliche Erläuterung im Kapitel 2.3.
- Streichen des Hinweises auf das Wortspiel. Ohne die ausführliche Erläuterung besagt es nichts. Die komplette Wiederholung aus Kapitel 2.3 macht keinen Sinn.

- Lizenzen mit reiner Patentnutzungszusage (z.B. European Public Licence 1.1)
- Lizenzen mit Patentnutzungszusagen und Androhung eines Wegfalls der Nutzungszusage bei einem Lizenzverstoß (z.B. Apache Software License 2.0, GPL V3).

Es liegt damit auf der Hand, dass der Einsatz von Open-Source-Bestandteilen auch aus dem Blickwinkel des Patentrechts geprüft werden muss. Insbesondere bei Open-Source-Lizenzen, die eine Patent-Abwehrklausel vorsehen, sollte die Verwendung der zugehörigen Software nicht ohne Kenntnis der Konsequenzen und einer entsprechenden Abwägung von Vor- und Nachteilen erfolgen. Näheres zum Patenrecht findet sich in Kapitel 6.5.

6.2.3 Creative Commons und artverwandte Lizenzen

Da „klassische“ Open-Source-Lizenzen auf das Schreiben von Source- und / oder Objektcode in Software-Projekten ausgerichtet sind, lassen sich ihre Inhalte nur schlecht auf andere Einsatzbereiche wie die Erstellung von Grafiken, Sounds, Fonts und ähnliche Werken übertragen. Aufgrund der Tatsache, dass die Community die Grundüberlegungen der Open-Source-Bewegung auch auf diese anderen digitalen Schöpfungen übertragen will, haben sich in diesem Bereich einige Lizenzen entwickelt, die diese Lücke schließen sollen.

Am bekanntesten und am weitesten verbreitet ist hier die Lizenzfamilie der Creative Commons Lizenzen, die für kreative Schöpfungen einen „Baukasten“ von verschiedenen Lizenzen anbietet. Auch hier lässt sich eine grobe Trennlinie zwischen Varianten ohne Copyleft (z.B. Creative Commons Attribution, abgekürzt CC-BY) oder solchen mit Copyleft (bei Creative Commons „ShareAlike“ genannt, abgekürzt z.B. CC-BY-SA) ziehen. Daneben gibt es noch Variationen, die z.B. eine rein nicht-kommerzielle Nutzung vorschreiben.

Eine weitere häufig genutzte Open Data Lizenz ist die Open Database License 1.0 (ODbL 1.0), die speziell für die Kartendaten des Open Street Map Projekts entworfen wurde, nachdem dort zunächst die Creative Commons Attribution-ShareAlike 2.0 verwendet wurde.

6.2.4 Multiple Lizenzierung

Urheber von Open-Source-Software müssen sich nicht auf eine einzige Lizenz für das von ihnen geschaffene Werk festlegen. Sie können vielmehr ein und dieselbe Software unter zwei (oder mehreren) unterschiedlichen Lizenzen verbreiten. Für eine solche „dual“- oder „multi“-Lizenzierung können sie auch völlig konträre Lizenzen und Lizenztypen wählen, z.B. eine kommerzielle Lizenz und eine Open-Source-Lizenz. Der Nutzer seinerseits kann sich die für ihn am besten passende Lizenz heraussuchen. Häufig wird für eine multiple Lizenzierung als Open-Source-Lizenz eine solche mit Copyleft-Effekt gewählt. Dies führt dazu, dass Interessenten, die beispielsweise eigene Modifikationen des Werks nicht veröffentlichen wollen, eher zur kommerziellen Lizenz greifen werden und damit den wirtschaftlichen Erfolg des Urhebers steigern. Oft richtet sich eine solche alternative Lizenzierung auch schlicht nach Praktikabilitäts- oder Kompatibilitätsgesichtspunkten. Beispielsweise wählen viele Urheber möglichst permissive Open-Source-Lizenzen (z.B. die MIT-Lizenz), wenn eine Einbindung in möglichst viele andere Open-Source- Komponenten bzw. Projekte ermöglicht werden soll.

Zu beachten ist, dass es neben der Möglichkeit einer alternativen Lizenzierung auch kumulative Lizenzierungen gibt. Darunter versteht man Fallgestaltungen, in denen zwei oder mehrere Lizenzen gleichzeitig gelten sollen. Es liegt auf der Hand, dass in solchen Fällen genauestens untersucht werden muss, ob die kumulativ kombinierten Lizenzen tatsächlich kompatibel sind. Wird hier vom Urheber unsauber gearbeitet, führt dies in der Regel dazu, dass das Werk aus urheberrechtlichen Gründen nicht einsetzbar ist.

Kann ein Verwender von Open-Source-Software zwischen mehreren Lizenzen wählen, so muss dies immer mit Blick auf die Lizenzkompatibilität auch zu den Bedingungen für den eigenen Code erfolgen. Anbieter von kommerzieller Software werden sich üblicherweise häufig für die Option mit den

permissivsten Bestimmungen entscheiden und Lizenzen ohne Copyleft-Effekt bevorzugen, wenn solche zur Auswahl stehen. Es sollte zumindest intern festgehalten werden, welche Lizenzierungsoption im jeweiligen Fall gewählt wurde, insbesondere, um die Einhaltung der jeweiligen Lizenzpflichten dokumentieren zu können. Ansonsten kann es im Nachhinein Unsicherheiten geben, welche Lizenz gilt und welche Lizenzpflichten eingehalten werden müssen.

6.3 Lizenzkompatibilität unterschiedlicher OSS-Lizenzen

Werden mehrere Softwarekomponenten zusammen genutzt oder sollen verschiedene Softwarekomponenten in einem Gesamtprodukt miteinander kombiniert und vertrieben werden, stellt sich die Frage, ob die jeweiligen Softwarelizenzen, die den einzelnen Paketen zugrunde liegen, miteinander kompatibel sind – oder anders gesagt, ob eine Nutzung der Softwarekomponenten ohne Verletzung jeweils einer der Lizenzen möglich ist. Das Thema der Lizenzkompatibilität spielt sowohl bei proprietärer Software als auch bei OSS eine Rolle, es wurde bislang allerdings vornehmlich im Rahmen von OSS-Lizenzen diskutiert.

Eine Verbindung von einzelnen Softwarekomponenten, deren Lizenzen keine Copyleft-Klausel enthalten, ist in vielen Fällen problemlos möglich. So können beispielsweise Komponenten, die unter einer sogenannten 2-clause BSD-Lizenz lizenziert sind, mit Komponenten verbunden werden, die unter einer MIT-Lizenz stehen.

Dagegen ergeben sich Probleme bei OSS-Lizenzen mit einem Copyleft-Effekt, wie der GPL oder der LGPL. Denn diese enthalten eine Verpflichtung, abgeleitete Werke ausschließlich unter ihren eigenen Bedingungen zu verbreiten. Diese Lizenzen sind dann nicht mit anderen Lizenzen kompatibel, wenn eine Copyleft-Lizenz einer Komponente eine Lizenzierung des Gesamtpaketes unter der Copyleft-Lizenz verlangt, die Lizenz einer anderen Komponente genau dies jedoch nicht erlaubt. Ein Beispiel ist etwa die statische Verlinkung einer GPLv2-lizenzierten Bibliothek mit einer proprietär lizenzierten Softwarekomponente, welche die Weitergabe des Source Code des Gesamtpaketes verbietet, was die GPLv2 wiederum gerade verlangt. Werden hingegen zwei OSS-Komponenten nicht so miteinander verbunden, dass ein Copyleft-Effekt greift, dann besteht kein Kompatibilitätsproblem. Dasselbe gilt, wenn zwar ein Copyleft-Effekt greift, eine Anwendung der Copyleft-Lizenz auf das Gesamtpaket jedoch ohne Pflichtenkollision mit übrigen Lizenzen möglich ist.

Lizenzinkompatibilitäten können sich auch aus der Kollision zweier Copyleft-Lizenzen ergeben, die eine Lizenzierung des Gesamtpaketes jeweils unter den eigenen – von der anderen Copyleft-Lizenz abweichenden Bedingungen verlangen, etwa bei der Kollision der GPLv2 mit der GPLv3.

Die Lizenzen der GPL-Familie enthalten den wohl strengsten Copyleft-Effekt (vgl. Ziffer 6 der GPLv2), so dass eine Kompatibilität mit anderen Lizenzen nur in wenigen Fällen gegeben ist.

Um Inkompatibilitäten zu vermeiden, beinhalten einige Copyleft-Lizenzen spezielle Öffnungs- oder Kompatibilitätsklauseln, welche die Nutzung von abgeleiteten Werken auch unter anderen Copyleft-Lizenzen erlauben (vgl. etwa Ziffer 3 der LGPLv2.1). Auch kommt in der Praxis die alternative Mehrfachlizenzierung desselben Programms unter verschiedenen Lizenzen vor (oft als Dual Licensing bezeichnet). Besteht ein solches Wahlrecht, so sollte eine mit dem jeweiligen Gesamtpaket kompatible Lizenz gewählt werden.

Eine Lizenzkompatibilität von Copyleft-Lizenzen mit anderen Lizenzen ist im Ergebnis also nur dann gegeben, wenn entweder durch eine Kompatibilitätsklausel in der jeweils anderen Lizenz auch die Nutzung unter den Lizenzbedingungen der Copyleft-Lizenz erlaubt wird oder wenn die andere Lizenz keine Verpflichtungen enthält, welche über die Lizenzpflichten der Copyleft-Lizenz hinausgehen oder diesen widersprechen.

Die Free Software Foundation hat umfangreiche Listen und Tabellen veröffentlicht, in welcher die Vereinbarkeit von Lizenzen mit Fokus auf die (L)GPL dargestellt wird.[38]

Allerdings lässt sich keine allgemeingültige Aussage zur Kompatibilität von Softwarelizenzen treffen. Dies würde nämlich voraussetzen, dass sämtliche Open-Source-Lizenzen widerspruchsfrei, unmissverständlich und eindeutig formuliert sind und dass sie von allen Beteiligten identisch interpretiert werden, was in der Praxis jedoch leider nicht der Fall ist. Viele Lizenzen lassen bezüglich einzelner Klauseln Raum für unterschiedliche Auffassungen und Interpretationen. Daher besteht in der Open-Source-Community in vielen Fällen Uneinigkeit zur Frage der (In-)Kompatibilität einzelner Lizenzen. Pauschale Aussagen zur Vereinbarkeit sind also mit Vorsicht zu genießen.

Letztlich kommt es bei der Frage der Lizenzkompatibilität auf den konkreten Einzelfall an, insbesondere auf die Frage, ob die betroffenen Komponenten auch getrennt vertrieben werden können oder ob beispielsweise eine Öffnungs- oder Kompatibilitätsklausel enthalten ist. Für die Beurteilung, wie weit der Copyleft-Effekt im Einzelfall reicht und ob Lizenzen miteinander kompatibel sind, spielen aber auch technische Kriterien eine wesentliche Rolle: Wird die Komponente statisch oder dynamisch verlinkt? Gibt es tiefergehende Funktionsaufrufe? Wird mit der Komponente nur auf Kommandozeilenebene kommuniziert?

Folglich muss nicht nur geklärt werden, welche Software eingesetzt wird, sondern auch, wie sie eingesetzt wird. Nur bei konkreten Informationen zur Einbindung der Software in das Gesamtpaket kann tatsächlich eine belastbare Aussage zum Copyleft-Effekt getroffen werden. Eine rechtliche Bewertung der einzelnen Parameter ist insofern unverzichtbar. Vor dem Hintergrund der Unklarheit zahlreicher Lizenztexte kommt es auch auf das Verständnis des jeweiligen Lizenzgebers vom Lizenztext an. Hat dieser etwa seine Auffassung über die Reichweite des Copyleft-Effekts in einer FAQ-Liste veröffentlicht, so ist diese Auffassung gegebenenfalls vorrangig vor der Interpretation etwa der Free Software Foundation.

Insbesondere bei Dual-Licensing-Modellen, die etwa die Verwendung von Komponenten wie Datenbank-Konnektoren in proprietärer Software nur im Fall des Erwerbs kostenpflichtiger Lizenzen erlauben, ist die Reichweite des Copyleft-Effekts und damit die Kompatibilität der OSS-Lizenz mit einer proprietären Lizenz besonders kritisch zu prüfen. Bestimmte Lizenzen blockieren sich wechselseitig und verhindern eine Kombination des zugrunde liegenden Codes. So können z.B. Dateien, die unter verschiedenen Lizenzen jeweils mit Copyleft-Effekt stehen, nicht in einem Open-Source-Paket (derivative work) kombiniert werden. Im Einzelfall kann man aber eine Kompatibilität herstellen (abhängig von der technischen Umsetzung).

6.4 Urheberrecht

6.4.1 Abgrenzung zum anglo-amerikanischen Copyright

Zum Verständnis der meisten OSS-Lizenzen ist es hilfreich, sich einige wesentliche Unterschiede zwischen dem anglo-amerikanischen Begriff „Copyright“ und dem kontinental-europäischen, hier vor allem deutschen, Begriff „Urheberrecht“ vor Augen zu führen:

Das deutsche Urheberrecht ist als Persönlichkeitsurheberrecht ausgestaltet (vgl. §§ 12 ff. UrhG). Es besteht aus einer persönlichkeits- und einer vermögensrechtlichen Komponente. Beide Komponenten sind nach deutschem Verständnis untrennbar miteinander verbunden. Demgegenüber gibt das „Copyright“ dem Urheber Rechte zur Vervielfältigung oder Verbreitung, mithin die hauptsächlich

[38] siehe http://www.gnu.org/licenses/license-list.html sowie http://www.gnu.org/licenses/gpl-faq.html#AllCompatibility

verwertungsbezogenen Rechte, die im deutschen Urheberrecht vor allem in §§ 15 ff. UrhG geregelt sind.

Dies hat zur Folge, dass das deutsche Urheberrecht als solches, im Gegensatz zum „Copyright", grundsätzlich nicht übertragen werden kann (§ 29 Abs. 1 UrhG). Stattdessen können einfache oder ausschließliche Nutzungsrechte eingeräumt werden, die so weit gehen können, dass der Urheber selbst von der Nutzung seines Werks ausgeschlossen ist. Diese Nutzungsrechte können auch nach deutschem Recht als übertragbare Rechte eingeräumt werden. Es können also nur Nutzungsrechte am Werk, nicht aber das Urheberrecht als solches übertragen werden. Die Einräumung bzw. Übertragung solcher Nutzungsrechte erfolgt allgemein durch Lizenzvertrag.

Der Begriff „Copyright" entspricht also nicht vollständig dem deutschen Begriff des „Urheberrechts". Wenn in OSS-Lizenzen, die in weit überwiegender Zahl in ihrer Systematik auf US-amerikanischem Lizenzrecht und damit dem „Copyright"-Gedanken beruhen, daher von der „Übertragung des Copyright" die Rede ist, ist dies nach deutschem Recht als Einräumung bzw. Übertragung von Nutzungsrechten anzusehen.

6.4.2 Rechtsinhaberschaft

Die Bestimmung der Rechtsinhaberschaft an einer OSS, d.h. die Identifizierung der Personen oder Organisationen, denen die Urheberrechte an der Software zustehen, kann sehr schwierig sein.

Grundsätzlich ist Urheber eines Werkes dessen Schöpfer. Wird eine Software als urheberrechtlich geschütztes Werk von mehreren Personen gemeinsam geschaffen, ohne dass sich ihre Beiträge gesondert verwerten lassen, so sind diese Personen Miturheber der Software (vgl. § 8 UrhG). Wesentliche Folge einer solchen Miturheberschaft ist, dass eine Verwertung der Rechte an dem Werk grundsätzlich nur mit vorheriger Zustimmung aller Miturheber möglich ist.

Bei komplexeren Werken oder ganzen „Distributionen", bei denen selbständige Programme oder Programmteile zunächst unabhängig voneinander entstanden sind und später miteinander verbunden werden, entsteht keine Miturheberschaft aller Beitragenden an dem gesamten Werk, da die einzelnen Programmteile gesondert verwertet werden können. Stattdessen entsteht ein sogenanntes „verbundenes Werk" i.S.d. § 9 UrhG. In diesem Fall bleibt jeder Programmierer (Allein-)Urheber des jeweils von ihm geschaffenen Moduls, wobei für Module auch Miturhebergemeinschaften denkbar sind. Somit können in einem „verbundenen Werk" Module einzelner Urheber und Miturhebergemeinschaften verbunden sein.

Prägend für den OSS-Gedanken ist schließlich die sukzessive Bearbeitung einer OSS-Komponente durch mehrere Urheber. Eine solche Bearbeitung eines Werks bedarf zwar grundsätzlich der Einwilligung des oder der Urheber (vgl. §§ 23, 69c Nr. 2 UrhG), sie ist bei OSS jedoch nachgerade erwünscht. Aus einer bestehenden Software kann somit im Lauf der Zeit eine u.U. erheblich umgestaltete neue Software werden. Soweit die jeweiligen Programmierer nicht nur unwesentliche Bearbeitungen vornehmen, haben sie nach § 3 UrhG ein Bearbeiterurheberrecht, welches sich allerdings ausschließlich auf die Bearbeitung bezieht.

Für Software, die von einem Arbeitnehmer in Wahrnehmung seiner Aufgaben oder nach den Anweisungen seines Arbeitgebers geschaffen wird, bleibt der Arbeitnehmer zwar (Mit-)Urheber der Software. Jedoch stehen seinem Arbeitgeber – sofern nichts anderes vereinbart ist – nach § 69b Abs. 1 UrhG alle vermögensrechtlichen Befugnisse und damit auch die ausschließlichen Nutzungsrechte an dieser Software zu. Die „Rechtsinhaberschaft" liegt somit, bis auf ein dem Programmierer verbleibendes ideelles Urheberrecht, beim Arbeitgeber. Dies lehnt sich an die im US-Urheberrecht geltende „Work Made For Hire-Doktrin" (17 USC §101) an, wonach dem Auftrag- oder Arbeitgeber grundsätzlich das „Copyright" am geschaffenen Werk zusteht.

Im Ergebnis lässt sich festhalten, dass die Urheberschaft an einer OSS Komponente je nach Anzahl der Autoren bzw. Urheber sehr einfach (ein Urheber) oder auch äußerst kompliziert (viele miteinander verbundene einzelne Urheber und Urhebergemeinschaften) strukturiert sein kann, was sich im Fall der

Bearbeitung durch Dritte noch steigert. Zwar erleichtert zumindest im deutschen Recht § 10 UrhG den Nachweis ein wenig dadurch, dass gesetzlich die Urheberschaft oder Rechtsinhaberschaft bei demjenigen vermutet wird, der für ein Werk als Urheber oder Herausgeber angegeben ist. Dennoch kann die Rechtsinhaberschafft an einer bestimmten OSS sehr schwer zu bestimmen oder nachzuweisen sein.

6.4.3 Rechteübertragung

Nach § 69c UrhG bedürfen bestimmte Handlungen im Zusammenhang mit Computerprogrammen grundsätzlich der Zustimmung des Rechteinhabers. Hierzu gehört insbesondere die dauerhafte oder vorübergehende Vervielfältigung des Computerprogramms, dessen Bearbeitung, Verbreitung oder öffentliche Zugänglichmachung. Der Rechtsinhaber kann anderen Personen durch Lizenz die Erlaubnis zur Vornahme solcher Handlungen erteilen. Urheber von OSS erteilen entsprechende Lizenzen nicht zur Erzielung von Lizenzentgelten, sondern zur Realisierung und Sicherung von Nutzungsmöglichkeiten. Insofern gestatten die Bedingungen von OSS-Lizenzen durchweg die Weiterverbreitung der OSS. Sie gestatten die Vervielfältigung und Nutzung, erlauben die Analyse und Bearbeitung bzw. Weiterentwicklung der Software und gewähren somit umfassende Nutzungsrechte. Einige OSS-Lizenzen, darunter die Lizenzen der GPL-Familie, regeln ausdrücklich, dass die Weiterverbreitung nur unentgeltlich erfolgen darf. In der Regel handelt es sich aus urheberrechtlicher Sicht bei den durch OSS-Lizenzen eingeräumten Rechten um „nicht ausschließliche Nutzungsrechte“ („non-exclusive right“). Dies bedeutet, dass die OSS auf die erlaubte Art genutzt werden darf, ohne dass eine Verwendung durch andere ausgeschlossen ist bzw. Dritten durch den Lizenznehmer untersagt werden kann.

Häufig sind in OSS-Lizenzen weitere Bedingungen enthalten, die nicht direkt an die Einräumung von Rechten geknüpft sind. So sehen die Open-Source-Kriterien der OSI ein Diskriminierungsverbot vor, wonach Open-Source-Lizenzen weder Nutzer noch Einsatzbereiche einschränken dürfen. OSS kann in solchen Fällen beispielsweise auch zur Entwicklung von Waffentechnologie oder in der Gentechnik eingesetzt werden.

Umstritten ist, ob für das bloße Ablaufenlassen der Software, beispielsweise auf einem Computer, eine Lizenz und die Akzeptanz entsprechender Lizenzbedingungen erforderlich ist. Bei dem Ablaufenlassen handelt es sich um eine „*bestimmungsgemäße Nutzung*“ im Sinne des § 69d Abs. 1 UrhG. Daher wird vertreten, dass durch § 69d Abs. 1 UrhG eine gesetzliche Erlaubnis gegeben ist, sodass das reine Ablaufenlassen keine zusätzliche vertragliche Vereinbarung der Lizenzbedingungen erfordert (vgl. insoweit auch der Text in Ziffer 0 der GPLv2: "*Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted (…)*"). Wird OSS dagegen über das bloße Ablaufenlassen hinaus genutzt, insbesondere verbreitet oder vertrieben („distribution“), greifen die jeweiligen Lizenzbedingungen.

Schließlich können die Urheber gewährte Nutzungsrechte an OSS auch wieder entziehen, wenn die Nutzungsbedingungen der OSS-Lizenz nicht eingehalten werden. So beinhaltet z.B. Ziffer 4 der GPL v2 eine Regelung, wonach die eingeräumten Rechte bei einem Verstoß gegen die Lizenz automatisch entfallen. Das LG München hat diese Klausel als wirksam angesehen.[39]

6.4.4 Beurteilung von OSS-Lizenzen nach deutschem Recht

Die meisten der für OSS verwendeten Lizenzbedingungen basieren in ihrer Systematik auf US-amerikanischem Lizenzrecht. Die Orientierung an der US-amerikanischen Systematik bedeutet allerdings nicht, dass

- automatisch amerikanisches Recht Anwendung findet oder

[39] vgl. LG München I, Urt. v. 19.5.2004, Az. 21 O 6123/04

- die Klauseln nach der deutschen Rechtsordnung automatisch unwirksam sind.

Vielmehr gilt auch für aus dem Ausland (z.B. aus dem US-amerikanischen Raum) stammende OSS das Schutzlandprinzip. Das bedeutet, es ist das Recht desjenigen Staates anzuwenden, für dessen Gebiet der Schutz beansprucht wird. Somit ist deutsches Urheberrecht anzuwenden, wenn z.B. US-amerikanische OSS in Deutschland verwendet wird. Die Urheber und Rechteinhaber genießen gegen Beeinträchtigungen ihrer Urheberrechte in Deutschland den Schutz des deutschen Urheberrechts.

Nach dem deutschen Urheberrecht gehören zu den geschützten Werken auch Computerprogramme, die in § 2 Abs. 1 Nr. 1 UrhG ausdrücklich als Beispiel für Sprachwerke genannt werden. Dass der deutsche Gesetzgeber darüber hinaus auch grundsätzlich von der Schutzmöglichkeit von Werken wie OSS ausgeht, zeigt sich z.B. in § 32 Abs. 3 Satz 3 UrhG. In dieser Regelung wird eine Ausnahme vom gesetzlichen Anspruch auf eine angemessene Vergütung für die Verwendung eines Werks gemacht: "*Der Urheber kann aber unentgeltlich ein einfaches Nutzungsrecht für jedermann einräumen.*" Der hinter OSS stehende Gedanke der freien und unentgeltlichen Verwendung findet somit auch im Gesetz eine Grundlage.

Andererseits muss sich die Wirksamkeit einzelner OSS-Lizenzbedingungen am deutschen Recht (u.a. am AGB-Recht, vgl. weiterführend hierzu Ziffer 6.7.1) messen lassen. Dass eine solche Vereinbarkeit nicht von vornherein ausgeschlossen ist, hat u.a. das LG München I bestätigt.[40] Das Gericht hatte sich mit der GPLv2 zu befassen und bejahte ohne Weiteres deren Eigenschaft als Allgemeine Geschäftsbedingungen i.S.d. § 305 BGB. In ihrem Urteil sahen die Münchener Richter die Rechterückfallklausel in Ziffer 4 der GPLv2 als mit deutschem AGB- und Urheberrecht vereinbar an.

6.4.5 Geltendmachung von Urheberrechten an Open Source Software

Wie unter 6.5.2 beschrieben ist OSS häufig ein Gemeinschaftswerk mit sehr komplexer Urheberstruktur. Oft kennen sich die Miturheber untereinander nicht oder stehen nur in losem Kontakt, z.B. per E-Mail oder Forenmitgliedschaft. Teilweise ist den (Mit-)Urhebern der wirkliche reale Name der anderen (Mit-)urheber nicht bekannt, sondern nur das Pseudonym. Dies führt zu Schwierigkeiten, wenn es darum geht, wegen tatsächlicher oder vermeintlicher Verletzungen der OSS-Lizenz rechtlich gegen den Verletzer vorzugehen.

Besteht eine Miturheberschaft, kann bei Verletzungen des (Mit-)Urheberrechts zwar jeder Miturheber Unterlassung verlangen bzw. Unterlassungsklage erheben. Jeder Miturheber kann also von einem unberechtigten Nutzer verlangen, dass dieser die Nutzung der OSS einstellt. Die Zustimmung der anderen Urheber ist hierfür nicht erforderlich (§ 8 Abs. 2 Satz 3 Halbsatz1 UrhG). Über das Urheberrecht verfügen kann die Miturhebergemeinschaft aber nur gemeinschaftlich (vgl. § 8 Abs. 2 Satz 1 UrhG). Dies hat zur Folge, dass ein Miturheber eine Leistung nur an alle Miturheber (z.B. im Rahmen einer Leistungsklage) verlangen kann. Er muss dazu also alle Miturheber benennen können.

Das Vorstehende gilt nach unstrittiger Ansicht entsprechend für die Urheber verbundener OSS-Werke, wobei dann entweder § 8 Abs. 2 UrhG analog oder § 744 Abs. 2 BGB direkt angewendet wird (sogenannte „Notverwaltung“ des gemeinschaftlichen Gegenstands).

Lediglich bei Bearbeitungen, an denen ein eigenes Urheberrecht des jeweiligen Urhebers (nur) an der Bearbeitung besteht, kann dieser Bearbeiterurheber seine Leistungsrechte an eben seiner Bearbeitung allein geltend machen. Eine Unterlassung kann jedoch im Sinne des § 8 Abs. 2 UrhG auch für die gesamte OSS verlangt werden.

Aus Sicht eines Rechtsverletzers kann dies zu der äußerst problematischen und riskanten Situation führen, dass ein einzelner Miturheber Unterlassung verlangen kann, mithin die betroffene OSS, die möglicherweise eine wichtige Komponente einer anderen Software darstellt, nicht mehr genutzt werden darf. Ohne Zusammenwirken aller Miturheber ist es jedoch nicht möglich, sich mit diesen über die Modalitäten einer Fortsetzung des Einsatzes der OSS zu einigen, etwa im Wege eines Vergleichs

[40] vgl. Urteil vom 19.5.2004 (Az. 21 O 6123/04)

oder einer Lizenzvereinbarung. Je nachdem wie einfach die betroffene OSS-Komponente zu ersetzen ist, kann dies erhebliche finanzielle Folgen für den Rechtsverletzer haben.

6.5 Patente und Marken

Das Urheberrecht bildet in Deutschland die wesentliche gesetzliche Grundlage für Software. Daneben können jedoch weitere gewerbliche Schutzrechte, z.B. Patent- und Markenrechte relevant werden.

6.5.1 Patentrecht

6.5.1.1 Gegenstand des Patentrechts

Das Patentrecht dient dem Schutz einer Erfindung, die auch als Lehre für ein technisches Handeln bezeichnet wird. Ein erteiltes Patent verleiht dem Anmelder oder Patentinhaber das Recht, anderen die gewerbliche Ausübung der patentierten Lehre zu verbieten. Hierdurch wird dem Anmelder eine zeitlich befristete Monopolstellung verliehen, wofür er im Gegenzug die technische Lehre offenlegen muss. Außerdem wird die technische Lehre nach Ablauf der Patentlaufzeit von maximal 20 Jahren gemeinfrei, d.h. jeder darf sie nutzen.

Im Patentrecht sind zwei unterschiedliche Anspruchskategorien zu beachten, die unterschiedliche Verbietungsgrade zu Folge haben, nämlich Vorrichtungsansprüche und Verfahrensansprüche. Ein Vorrichtungsanspruch ist auf einen physisch greifbaren Gegenstand gerichtet, wie beispielsweise ein Gerät, eine Schaltung, eine chemische Substanz. Bei Vorliegen eines validen Vorrichtungsanspruchs ist einem Dritten eine Mehrzahl von Handlungen, beispielsweise das Herstellen und sogar das Besitzen, verboten. Ein Verfahrensanspruch hingegen bewirkt im Wesentlichen nur das Verbot des Anwendens und Anbietens. Weitere verbotene Handlungen sind in § 9 des Patentgesetzes angegeben.

Bei industrieller Software wird es in der Regel eher um Verfahrenspatente gehen. Allerdings ist ein Patentanspruch auf ein Computerprogrammprodukt (vgl. die gleichnamige Entscheidung einer technischen Beschwerdekammer des EPA in einem Erteilungsverfahren[41]) oder ein Patentanspruch auf ein Gerät mit einer embedded Software (die eine computer-implementierte Erfindung realisiert) rechtlich wie eine Vorrichtung zu behandeln.

Eine Patentverletzung kann – vereinfacht gesagt – vorliegen, wenn im gewerblichen Bereich eine patentrechtlich geschützte Lehre ohne Zustimmung des Patentinhabers verwendet wird. Der gesetzliche Wortlaut ist dem Patentgesetz (§§ 9 bis 13 PatG) zu entnehmen.

Kommentiert [KT2]: Kommentar Kriesel: Dieser Abschnitt wurde aus 6.5.1.3 vorgezogen.

6.5.1.2 Patentschutz für Open-Source-Software

Ein Patent wird auf Antrag für eine technische Lehre erteilt, die neu ist, auf einer erfinderischen Tätigkeit beruht und gewerblich anwendbar ist. In Deutschland und vor dem europäischen Patentamt ist Software „als solche“ nach derzeitiger Gesetzeslage nicht patentfähig (§ 1 des deutschen Patentgesetzes (PatG) und Art. 54 des Europäischen Patentübereinkommens (EPÜ)), da ihr das Patentfähigkeitskriterium der technischen Lehre fehlt. In den USA ist ein Patentanspruch auf ein Programm „per se“ gemäß aktuellem MPEP 2106[42] unter Verweis auf die US-Rechtsprechung nicht zulässig.

Falls aber eine Software ein Verfahren implementiert, das technischer Natur ist, so kann diese Software möglicherweise patentrechtlich relevant sein. Die Rechtsprechung sieht eine Patentfähigkeit als gegeben an, sofern die Erfindung nicht nur neu, erfinderisch und gewerblich anwendbar ist,

[41] T 1173/97 Computerprogrammprodukt
[42] MPEP = Manual of Patent Examining Procedure, Handbuch für die Prüfer des US-amerikanischen Patentamts, herunterladbar unter www.uspto.gov

sondern zusätzlich als sogenannte computer-implementierte Erfindung über das übliche Zusammenwirken mit einem Rechner hinaus einen technischen Inhalt aufweist, bzw. durch diese Erfindung unmittelbar ein technischer Effekt ausgelöst wird.[43]

Sofern also ein Unternehmen beispielsweise in dieser Reihenfolge (!) eine Patentanmeldung auf eine technische Lehre einreicht, diese Lehre in einer Software implementiert und diese Software hernach vertreibt, so genießt dieses Unternehmen für Produkte, in denen diese technische Lehre realisiert ist, patentrechtlichen Schutz, abhängig vom Status der Patentanmeldung. Daran ändert sich auch nichts, wenn das Unternehmen diese Software mitsamt Quellcode unter einer Open-Source- Lizenz veröffentlicht.

Wird die Reihenfolge derart abgewandelt, dass die Software zunächst veröffentlicht wird und erst hernach die Patentanmeldung eingereicht wird, so wird die veröffentlichte Software der Patenterteilung oder der Aufrechterhaltung des Patents entgegenstehen, weil die der veröffentlichten Software entnehmbare technische Lehre der Patentanmeldung die Neuheit nehmen kann.

Mit anderen Worten: Wird eine Software veröffentlicht und kann ein Fachmann die technische Lehre dem mitgelieferten Quellcode unzweideutig entnehmen, so kann es sein, dass diese technische Lehre (beispielsweise wegen mangelnder Neuheit) nicht mehr patentfähig ist, beziehungsweise ein darauf erteiltes Patent widerrufbar wäre.

Da der Begriff „Software“ aus patentrechtlicher Sicht mehrdeutig aufgefasst werden kann, wird zur begrifflichen Klarstellung das folgende Zitat vom europäischen Patentamt wiedergegeben:

The term "software" is considered to be ambiguous, because it may refer to a program listing written in a programming language to implement an algorithm, but also to binary code loaded in a computer-based apparatus, and it may also encompass the accompanying documentation. So in place of this ambiguous term the concept of a computer-implemented invention has been introduced.[44]

Vor diesem Hintergrund wird im Folgenden von einer computer-implementierten Erfindung gesprochen, wenn eine Software gemeint ist, deren Verfahrensschritte es ermöglichen, eine technische Lehre auszuführen.

Was es bedeutet, dass ein Programm eine technische Lehre ausführt, oder, was Software als solche ist, lässt sich aufgrund der Vielzahl deutscher und europäischer Rechtsprechung[45] in diesem Rahmen nicht darstellen. Für konkrete Einzelfragen hierzu wird auf Fachleute auf dem Gebiet des gewerblichen Rechtsschutzes verwiesen.

Der Patentschutz (einer computerimplementierten Erfindung) wird immer ergänzt durch den Urheberrechtsschutz der Software „als solche“. Das Urheberrecht regelt unter anderem das Recht zur Vervielfältigung, der Bearbeitung und der Verbreitung des Quellcodes (vgl. Ziffer 6.4).

6.5.1.3 Patente und Open-Source-Lizenzen

In der **Apache License 2.0 (ASL 2.0)** ist eine explizite Regelung zur Patentlizenz wie folgt enthalten:

Kommentiert [KT3]: Kommentar Kriesel: Was genau ist eine Patentlizenz (im Unterschied zu einer Software-Lizenz)? Der Begriff der Patentlizenz wird leider nirgendwo erläutert.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent

[43] Benkard: Patentgesetz, § 1 PatG, Rdz. 107, 108; 10. Aufl.; Verlag C. H. Beck, München 2006
[44] http://www.epo.org/news-issues/issues/software.html
[45] Als Beispiele aus der jüngeren deutschen Rechtsprechung mögen BGH X ZR 121/09 – Webseitenanzeige, BGH Xa ZB 20/08 Dynamische Dokumentengenerierung, BGH Xa ZR 54/06 Proxyserversystem, BGH X ZR 27/12 Fahrzeugnavigationssystem, dienen.

litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

Als Ausgangsfall sei nachstehend die Einbindung eines OSS-Programms in die Gerätesteuerung einer Produktionsanlage angenommen. Das übrige Programm dieser Steuerung erstellt der Hersteller der Produktionsanlage selbst und soll nur in Verbindung mit dieser Produktionsanlage verwendet und verkauft werden, also proprietär sein.

Fallgestaltungen:

a) Unveränderte Weitergabe
Annahme: Ein Unternehmen lädt sich die Software unter der ASL 2.0 herunter und gibt diese unverändert in einem Gerät weiter.

Folge: In diesem Fall hat der Contributor oder Licensor (diese Begriffe sind in der ASL 2.0 definiert) auf die Durchsetzung möglicherweise vorhandener eigener Patente verzichtet.

Da das betrachtete Unternehmen die Software unverändert weitergibt, dürften in der Regel unternehmenseigene Patente nicht betroffen sein, das Unternehmen selbst würde also keine Patentlizenzen an die Abnehmer und möglicherweise weitere Dritte vergeben müssen.

Ob allerdings aufgrund der (gewerblichen) Weitergabe durch das betrachtete Unternehmen ein Patent Dritter verletzt wird, kann nicht ausgeschlossen werden. In diesem Zusammenhang ist darauf hinzuweisen, dass es in der Verantwortung des gewerblichen Nutzers – also hier: des betrachteten Unternehmens – steht, zu überprüfen, ob sein Produkt in Schutzrechte Dritter eingreift. Dies gilt insbesondere dann, wenn z.B. ein Konkurrent des betrachteten Unternehmens, der ähnliche Produktionsanlagen herstellt, die mit der OSS durchgeführten Verfahrensschritte im Rahmen eines Verfahrens für diese Produktionsanlage patentiert hat. Wenn das Unternehmen, welches die Software heruntergeladen und weitergegeben hat, seinerseits ein Patent auf ein solches Verfahren anmeldet, so ist ein mit dieser OSS durchgeführter Verfahrensschritt alleine nicht geeignet, Neuheit und erfinderische Tätigkeit zu begründen, weil zum Zeitpunkt des Herunterladens die Open-Source-Software schon der Öffentlichkeit zugänglich war und damit nicht neu ist, wenn das Patent zeitlich nach dem Veröffentlichen der OSS angemeldet wird.

b) Veränderte Weitergabe mit eigenem Patent oder Patenten
Annahme: Ein Gerätehersteller lädt sich Software unter der ASL 2.0 herunter, nimmt daran Änderungen vor und gibt die daraus entstandene neue Software in einem Gerät weiter. Die Änderungen stellen eine computer-implementierte Erfindung dar und stehen vollständig unter einem oder mehreren unternehmenseigenen erteilten Patenten, die hier der Einfachheit halber als rechtsbeständig angenommen werden.

Folge: Hier gilt die gleiche Rechtsfolgewie unter a). Zusätzlich stellt das Unternehmen seinen Kunden frei, die Erfindung im Gerät zu nutzen.

Der Kunde ist weiterhin berechtigt, die unter der ASL 2.0 stehende Software seinerseits weiter zu verbreiten (vergleiche Punkt „4. Redistribution“ der Lizenz), falls der Gerätehersteller dies nicht beschränkt hat (zu einer solchen Beschränkung wäre er nach Punkt „5. Submission to Contribution“ der ASL 2.0 berechtigt).

c) Beitrag (Contribution) mit eigenem Patent oder Patenten
Annahme: Ein Unternehmen oder Entwickler lädt sich Software unter der ASL 2.0 herunter, nimmt daran Änderungen vor und gibt die daraus entstandene neue Software einschließlich Quellen als Beitrag an die Community zurück. Die Änderungen stellen eine computer-implementierte Erfindung

dar und stehen vollständig unter einem oder mehreren unternehmenseigenen Patenten, die hier der Einfachheit halber als rechtsbeständig angenommen werden.

Folgen: In diesem Falle verzichten das Unternehmen oder der Entwickler auf die Durchsetzung der eigenen Patente gegenüber nachfolgenden Lizenznehmern. Im unter a) angegebenen Beispiel ist also auch der Konkurrent berechtigt, den mit OSS durchgeführten Verfahrensschritt durchzuführen, vorausgesetzt er verwendet die entsprechende Software mit dem Beitrag des Unternehmens und erhält damit eine Nutzungs- und Patent-Lizenz daran. Hier ist jedoch Vorsicht geboten. Denn die ASL 2.0 versteht unter einem Unternehmen eine „Legal Entity" und meint damit die Einheit sämtlicher verbundener Unternehmen eines Konzerns. Verzichtet ein Unternehmen eines Konzerns auf die Durchsetzung von Patentrechten, gilt dieser Verzicht für alle im Konzern verbundenen Unternehmen.

Hinweise auf gegebenenfalls existierende Patentrechte Dritter bei Open-Source-Software sind in jedem Falle zu berücksichtigen.

Es ist weiterhin in jedem Fall darauf zu achten, dass die jeweiligen in der OSS-Lizenz festgelegten Lizenzbedingungen eingehalten werden, also ist beispielsweise der Lizenztext der jeweiligen Software beizufügen. Wird dies nicht beachtet, so folgt gemäß verschiedener OSS-Lizenzen der Verlust der Nutzungserlaubnis für die Software und einhergehend damit möglicherweise auch der Verlust jeglicher Patentlizenzen.

Ferner ist in der Fallgestaltung, in der ein Unternehmen im deutschen Rechtsraum eine Weitergabe von Software, die eine computer-implementierte Erfindung beinhaltet, vornimmt, dringend zu empfehlen, dass dieses Unternehmen mit seinem Erfinder eine Einigung im Sinne des Arbeitnehmererfindergesetzes erzielen möge, da der Vergütungsanspruch des Erfinders gegenüber dem Unternehmen durch den kostenfreien Beitrag der veränderten Software an die Community wesentlich berührt ist.

In der **Mozilla Public License 1.1 (MPL 1.1)**[46] ist die Wortwahl „Patents Claims infringed by ..." nicht so klar wie in der ASL 2.0 und scheint in dieser Form dem deutschen Recht eher fremd, denn „infringed by" impliziert in der hier vorliegenden Formulierung, dass eine Patent verletzende Handlung ohne die Zustimmung des Patentinhaber bereits begangen wurde. Genau hier aber wird ja gerade erst die Zustimmung durch den Patentinhaber erteilt und auch gerade erst die ein Patent betreffende Software weitergegeben. In pragmatischer Auslegung wird hier allerdings davon ausgegangen, dass der Initial Developer bzw. der Contributor Patentinhaber des oder der betreffenden Patente sind.

Die MPL 1.1 regelt in den Punkten 2.1 und 2.2, dass ein Initial Developer oder ein Contributor dem Verwender der Software eine Patentlizenz zur Nutzung der Software bzw. der modifizierten Software einräumen. Die Lizenz ist nur in Verbindung mit der Software wirksam. In der Regelung hinsichtlich der betroffenen Patentansprüche („Patents Claims infringed by ...") scheint der Ausdruck „Licensable by Initial Developer" bzw. „Licensable by Contributor" zu fehlen, andererseits macht die Regelung ohne diese Ausdrücke anscheinend keinen Sinn. Daher wurde hier pragmatisch davon ausgegangen, dass nur die Patentansprüche gemeint sind, über die der Initial Developer bzw. der Contributor verfügen können. Es werden in der MPL 1.1 verschiedene weitere Nutzungsvarianten geregelt, deren Behandlung nur in einer Einzelfallbetrachtung zweckmäßig ist. Eine solche Einzelfallbetrachtung kann hier nicht vorgenommen werden.

In ihrem Punkt 3.4 (a) regelt die MPL 1.1, dass ein Contributor, der Kenntnis davon hat, dass eine Lizenz eines gewerblichen Schutzrechtes eines Dritten zur Nutzung der Software erforderlich ist, in einer Datei namens „LEGAL" darauf hinzuweisen hat. Unabhängig von den Aussagen in einer Lizenz oder in begleitenden Unterlagen hat ein Softwarenutzer natürlich auch selbsttätig darauf zu achten, keine Schutzrechte Dritter zu verletzen.

Die MPL 1.1 enthält keine Patent Retaliation Clause (vgl. dazu 6.5.1.5).

[46] Der Lizenztext ist z.B. zu finden unter https://www.mozilla.org/en-US/MPL/1.1/

Kommentiert [KT4]: Kommentar Herr Teichert: m.E. ist hier ein Hinweis auf die Arbeitstitel „Konzernklausel" der Apache-2.0 Lizenz unabdingbar.
Siehe auch meinen Wunsch den OSS-Lizenznehmer unter 6.2 zu definieren, soweit eben möglich.
Das Verschweigen, dass es beim Patentnutzungsfreigeben um den ganzen Konzern geht und auch Klagen anderer Konzerngesellschaften hier relevant sind, könnte m.E. zu Schadensersatzansprüchen führen.
Das möchte ich vermeiden.

Kommentiert [KT5]: Kommentar Herr Teichert: Der „Konzern" ! Siehe vorherige Bemerkung.

Kommentiert [KT6]: Kommentar Kriesel: Diesen Verweis verstehe ich nicht. Ist d) gemeint?

Kommentiert [KT7]: Kommentar Kriesel: Ist das so richtig? Leider habe ich keine Ahnung von Patentrecht!

Kommentiert [KT8]: Kommentar Kriesel: Aufgrund des vorstehenden Kommentars von Herrn Teichert gestrichen.

Kommentiert [KT9]: Kommentar Kriesel: Textpassage aufgrund des vorstehende Kommentars von Herrn Teichert gestrichen.

Kommentiert [KT10]: Kommentar Kriesel: Diese Ausführungen verstehe ich nicht. Ist mit den Klauseln 2.1 und 2.2 der MPL 1.1 gemeint, dass der Lizenzgeber Handlungen erlaubt (z.B. Vervielfältigung, Nutzung), die ohne diese Erlaubnis Patentrechte verletzen würden? Ein solcher Gedanke wäre doch auch dem deutschen Recht nicht völlig fremd, oder?

6.5.1.4 Gemeinsamkeiten zwischen Open-Source-Lizenzen mit Copyleft und Patenten

Die Regelsysteme für Open-Source-Software und Patente haben mehr gemeinsam als es auf den ersten Blick scheinen mag: In beiden Systemen besteht eine Pflicht zur Offenlegung und in beiden Systemen besteht eine Pflicht zur Freigabe an die Allgemeinheit. Allerdings „hinkt“ das Patentrecht ein wenig hinterher: Eine Patentanmeldung und damit die technische Lehre wird 18 Monate nach dem Tag der Erstanmeldung offengelegt. Die technische Lehre, die durch das Patent geschützt wird, wird spätestens nach Ablauf des Schutzrechtes, das sind in der Regel maximal 20 Jahre ab Anmeldetag, zum Allgemeingut.

Kommentiert [KT11]: Kommentar Herr Teichert: Sonst ist die folgende Aussage falsch.
Vergleiche etwa BSD Lizenzen.

6.5.1.5 Patentverletzung und Retaliation (Patentabwehrklausel)

Kommentiert [KT12]: Kommentar Herr Teichert: Patentabwehrklausel wäre ein freundlicheres Wort

Das deutsche Patentrecht enthält keine Sonderregeln in Bezug auf Patente, die computer-implementierte Erfindungen zum Gegenstand haben. Auch für den Fall, dass eine computer-implementierte Erfindung durch eine Software implementiert ist, die unter einer Open-Source-Lizenz steht, gibt es keine Sonderregelungen.

Allerdings wäre im Einzelfall auf den Regelungsgehalt der Open-Source-Lizenz abzustellen, unter welcher eine Open-Source-Software steht. So findet sich in Open-Source-Lizenzen des Öfteren eine sogenannte „Retaliation“- oder Patentabwehrklausel. Die ASL 2.0[47] enthält eine solche Klausel in Abschnitt 3, Satz 2. Sie lautet in deutscher Übersetzung: Wer eine Open-Source-Software unter der ASL 2.0 nutzt und eine Patentverletzungsklage erhebt, für den erlöschen sämtliche Patentlizenzen, die er zur Ausführung dieser Open-Source-Software benötigt und die ihm im Rahmen der ASL 2.0 gewährt wurden. Die Wirkung dieser Klausel im Konzern ist umfassend: die Lizenz erlischt für den gesamten Konzern, auch wenn nur eine Konzerngesellschaft Klage erhoben hat.

Kommentiert [KT13]: Kommentar Herr Teichert: Auch hier ist m.E. ein Hinweis auf die Arbeitstitel „Konzernklausel“ der Apache-2.0 Lizenz erforderlich, denn auch die Klage einer Konzernschwester kann diese Wirkung auslösen.
Das Verschweigen, dass auch Klagen anderer Konzerngesellschaften hier relevant sind, könnte m.E. zu Schadensersatzansprüchen führen.
Das möchte ich vermeiden.

Diese Fallgestaltung ist präzisierend ein wenig zu ergänzen: Die Patentverletzungsklage kann nur ein Patentinhaber einreichen. Beklagter können der Licensor, der Contributor (sofern anwendbar) oder sonst jemand sein, welcher die Software in Verkehr bringt. Der die betrachtete Open-Source-Software nutzende Patentinhaber schadet sich also nur dann selbst, wenn diese Open-Source-Software noch wenigstens eine weitere patentierte computer-implementierte Erfindung eines Dritten implementiert – diese darf der klagende Patentinhaber dann nämlich gemäß ASL 2.0 nicht mehr nutzen, so dass er auch die betrachtete Open-Source-Software nicht mehr gewerblich nutzen darf.

Ist das Streitpatent des klagenden Patentinhabers allerdings das einzige Patent, dass von der betrachteten Open-Source-Software betroffen ist, so darf er gemäß ASL 2.0 die Software dennoch weiter nutzen, da die ASL 2.0 kein generelles Nutzungsverbot für den Kläger, sondern nur den Wegfall der Patentlizenz anordnet. Da der Kläger selbst Patentinhaber ist, stört ihn dies nicht; denn er benötigt keine Patentlizenz. Die Patentretaliation-Clause der ASL 2.0, also ASL 2.0 Abschnitt 3 Satz 2, scheint somit eher einen geringen Einfluss zu haben.

Kommentiert [KT14]: Kommentar Herr Teichert: Da es auch OSS Lizenzen gibt, bei denen nicht nur die Patentnutzung, sondern auch die Copyrightnutzung entzogen wird, soweit ich diese OSS Lizenzen verstehe, ist hier die Apache-2.0 ein etwas verharmlosendes Beispiel.
Ein Beispiel dazu wäre in Sektion 6.2 der CDDL-1.0 nachzulesen.
Mehr dazu hat der BITKOM AK OSS in einer Exceldatei.

Die Patentabwehrklausel ist in OSS-Lizenzen aber mit sehr unterschiedlicher Tragweite ausgestaltet. Es gibt Lizenzen, bei denen im Falle eines Patentstreites nicht nur die Patentlizenzen verloren gehen, sondern auch die urheberrechtlichen Lizenzen. Als Beispiel sei die CDDL genannt.

Auch für das Eingreifen einer Patentabwehrklausel gibt es sehr strenge Lizenzen, die nicht nur an die Erhebung einer Patentklage gegen die jeweilige OSS anknüpfen. So findet sich in der CPL eine Reaktion auf jede Art von Softwarepatent-Klage gegen einen der Autoren. Und im Fall der MPL löst sogar eine Hardwarepatent-Klage gegen einen der Autoren den Abwehrmechanismus aus. Interessant ist, dass für beide Lizenzen eine Nachfolge-Lizenz bzw. eine abgewandelte Lizenz mit weniger strengen Abwehrklauseln geschaffen wurde, nämlich EPL für die CPL und CDDL für die MPL.

[47] Vgl. z.B. http://www.apache.org/licenses/LICENSE-2.0.html

Im deutschen Rechtsraum ist eine Patentverletzungsklage in dieser Konstellation noch nicht breit publik geworden, falls es eine derartige Klage überhaupt schon gegeben hat.

Kommentiert [KT15]: Kommentar Frau Schnizer: Folgendes müßte erwähnt werden, um die Nutzer zu sensibilisieren. Rücksprache mit dem Autor (Hr. Block?) erforderlich.

Die Patent-Vergeltungsklausel ist in den OSS-Lizenzen mit sehr unterschiedlicher Tragweite ausgestaltet. Es gibt Lizenzen, bei denen im Falle eines Patentstreites nicht nur die Patentlizenzen verloren gehen, sondern auch die urheberrechtlichen Lizenzen. Als Beispiel sei die CDDL genannt.
Auch bezüglich des Eintritts der Patent-Vergeltungsklausel gibt es sehr strenge Lizenzen, in dem nicht nur eine Patentklage wegen Verletzung durch die jeweilige OSS vergolten wird. So findet sich in der CPL eine Vergeltung gegen jede Art von SW-Patentklage gegen einen der Autoren, und im Fall der MPL sogar zusätzlich eine Vergeltung im Falle einer HW-Patent-Klage gegen einen der Autoren. Interessant ist, dass für beide Lizenzen eine Nachfolge-Lizenz bzw. eine abgewandelte Lizenz mit weniger strengen Vergeltungsklauseln geschaffen wurde, nämlich EPL für die CPL und CDDL für die MPL.

6.5.1.6 Rechtsbeständigkeit von Patenten

Gegen ein zu Unrecht erteiltes Patent kann während einer Einspruchsfrist (in Deutschland und vor dem europäischen Patentamt 9 Monate nach seiner Erteilung) Einspruch eingelegt werden. Es wird widerrufen, wenn die zuständigen Spruchkörper es ebenfalls als zu Unrecht erteilt ansehen. Nach Ablauf der Einspruchsfrist kann der Widerruf nur noch über eine erheblich teurere Nichtigkeitsklage erreicht werden, sofern – für Deutschland oder mit Wirkung für Deutschland erteilte europäische Patente – das Bundespatentgericht zu dem Schluss kommt, dass das Patent zu Unrecht erteilt worden sei.

In den USA ist seit dem in mehreren Teilpaketen in Kraft getretenen America Invents Act (AIA) ein sogenanntes Review möglich, das vergleichbar zum deutschen oder europäischen Einspruch ist. Ferner hat es schon zuvor die Möglichkeit einer sogenannten Reexamination gegeben. In dieser Hinsicht wird kein Unterschied gemacht, ob bei dem Patent, das eine computer-implementierte Erfindung schützt, Open-Source-Software oder Closed Source betroffen ist.

6.5.2 Anmerkung zum Gebrauchsmusterrecht

Ein Gebrauchsmuster weist materielle und formelle Ähnlichkeiten, aber auch Unterschiede zum Patent auf und ist daher aus Sorgfaltsgründen als ein Schutzrecht Dritter zu beachten. Durch das Gebrauchsmusterrecht können nur Vorrichtungen, aber keine Verfahren geschützt werden, was es zur Anwendung im Bereich der computer-implementierten Erfindungen in der Praxis unattraktiver macht.

Auf die Ähnlichkeiten und Unterschiede soll hier nicht eingegangen werden, da ein Gebrauchsmuster hinsichtlich Schutzgegenstand und möglicherweise eintretender Rechtsfolgen mit einem Patent vergleichbar ist.

In Bezug auf Software, Open-Source-Software und computer-implementierten Erfindungen wird daher auf das zuvor zum Thema Patente Gesagte verwiesen.

6.5.3 Markenrecht

Das deutsche Markenrecht enthält keine spezifischen Regeln für den Umgang mit Open-Source-Software. Bei Verwendung und / oder Weitergabe einer Open-Source-Software sind daher die markenrechtlichen Vorgaben der Lizenz zu beachten, unter der die betrachtete Open-Source-Software heruntergeladen wurde. Hier wird im konkreten Fall eine einzelfallabhängige Betrachtung erforderlich sein.

Allerdings ist zu beachten, dass Nutzung und Einsatz einer Open-Source-Software nicht die Nutzung der Marke des Software-Urhebers einschließt. Eine OSS kann also nur dann in rechtlich unbedenklicher Weise verändert und in veränderter Form weitergegeben werden, wenn sämtliche Hinweise auf geschützte Marken sonstiger Mitautoren der Software gelöscht werden. Etwas anderes gilt nur, wenn in den Lizenzbedingungen der OSS die Verwendung von Marken der Urheber ausdrücklich gestattet wird. Dies ist aber regelmäßig nicht der Fall. Insofern können Marken von Herstellern von Open-Source-Software dazu verwendet werden, eigene Ausgestaltungen einer Software-Version besonders zu schützen.

Ein prominentes Beispiel ist „**Red Hat Enterprise Linux“, kurz „RHEL**“. Auch wenn alle Software-Bestandteile der GPL unterliegen, leitet Red Hat aus der tief in die Distribution eingebauten Marke bestimmte Rechte ab, wenn es um Lizenzierung und Weiterverbreitung der Software geht. Dies dient natürlich dem Schutz vor direkten Nachahmern aber auch der sauberen Abgrenzbarkeit von

vertragsrechtlichen Folgen wie Haftung, Gewährleistung und Freistellungsverpflichtungen in RHEL-Kaufverträgen. Um als Mitglied der Open-Source-Community gleichzeitig glaubwürdig zu bleiben, liefert Red Hat eine Anleitung, wie man die Marke „Red Hat“ aus RHEL entfernt, um bei der Erstellung von Quellcode-Kopien keine Markenrechtsverletzung zu begehen.

Generell hat ein Markeninhaber eine mögliche Verletzung seiner Marke zu überwachen und gegebenenfalls daraus vorzugehen, da ansonsten seine Ansprüche verwirkt sein können, wenn er die Benutzung der Marke während eines Zeitraums von fünf aufeinanderfolgenden Jahren in Kenntnis dieser Benutzung geduldet hat (§ 21 Markengesetz).

Weitere Beispiele: Eingetragene Marken, die zur Vermarktung von Waren und Dienstleistungen eines Open-Source-Programms verwendet werden, können auch gegen die Verwendung desselben Zeichens in einem eher proprietären Umfeld verteidigt werden. So konnten sich die Marken „Samba“ und „Gnome“ gegen den Zugriff anderer Interessenten behaupten.[48]

6.5.4 Designrecht

Das deutsche Designrecht enthält keine spezifischen Regeln für Open-Source-Software. Allerdings kann beispielsweise die Gestaltung einer Oberfläche (GUI oder GUI-Elemente) designrechtliche Relevanz haben. Bei Verwendung und / oder Weitergabe einer Open-Source-Software werden daher die designrechtlichen Vorgaben zu beachten sein, die in der Lizenz stehen, unter der die betrachtete Open-Source-Software heruntergeladen wurde. Hier wird im konkreten Fall eine einzelfallabhängige Betrachtung erforderlich sein. Außerdem hat auch hier der Unternehmer die Schutzrechte Dritter zu beachten. Da bereits das Design eines Buttons (einer Schaltfläche) eintragungsfähig sein kann, ist eine Recherche, beispielsweise in den Registereintragungen von designstarken Unternehmen, anzuraten.

6.5.5 Arbeitnehmererfindungen im Patentrecht

Das Recht der Arbeitnehmererfindung ist in Deutschland in weltweit einzigartiger Weise fein geregelt: Ein Arbeitnehmer, der eine Erfindung macht, hat diese seinem Arbeitgeber zu melden. Der Arbeitgeber hat diese Erfindung in Anspruch zu nehmen, was per Gesetz nach 4 Monaten als erfolgt gilt, oder die Erfindung dem Arbeitnehmer explizit freizugeben.

Im Gegenzug für die Inanspruchnahme ist der Arbeitgeber verpflichtet, die Erfindung als Schutzrecht beim Patentamt anzumelden. Umgekehrt wiederum hat der Arbeitnehmer einen Anspruch auf eine angemessene Vergütung für die Inanspruchnahme. Die Berechnung der Höhe der Vergütung ist in den Vergütungsrichtlinien festgelegt und hängt insbesondere vom Umsatz ab, der mit dem oder den Produkten erzielt wurde, in denen das Patent umgesetzt ist. Eine Vorab-Abbedingbarkeit ist per Gesetz ausgeschlossen.

Führt ein Arbeitgeber eine Erfindung in einem bestimmten Land nicht weiter, so hat er dem Arbeitnehmer die Weiterverfolgung des Schutzrechts dort anzubieten („Auslandsfreigabe“).

Die Regelungen nach dem Arbeitnehmererfinderrecht weichen demnach in erheblicher Weise von denen des Urheberrechts ab, wonach in der Regel dem Arbeitgeber das Recht an der Schöpfung automatisch zusteht und als mit dem Gehalt abgegolten gilt.

Fraglich ist, in wieweit eine Programmierleistung erfinderrechtlich relevant ist. Die Rechtsprechung sieht eine Patentfähigkeit als gegeben an, sofern die Erfindung nicht nur neu, erfinderisch und gewerblich anwendbar ist, sondern zusätzlich als sogenannte computer-implementierte Erfindung

[48] Die Beispiele sind beschrieben unter http://www.heise.de/ix/meldung/Telekom-zieht-Samba-als-Produktnamen-zurueck-2428412.html bzw. unter http://www.heise.de/open/meldung/Groupon-verzichtet-auf-den-Namen-Gnome-als-Marke-2452060.html.

über das übliche Zusammenwirken mit einem Rechner hinaus einen technischen Inhalt aufweist, bzw. durch diese Erfindung unmittelbar ein technischer Effekt ausgelöst wird. Als Rule-of-Thumbs mag dienen, dass beispielsweise eine Bearbeitung eines Datenbankskripts, ein Bau eines Parsers, ein Gestalten einer Benutzeroberfläche in der Regel (außer möglicherweise in einem besonders gelagerten Einzelfall) dem Patentschutz nicht zugänglich sind, so dass hier kein arbeitnehmererfinderrechtlich relevanter Beitrag zu vermuten ist. Anders kann der Fall bei Verfahren in der Telekommunikation oder bei kryptographischen Protokollen liegen.

6.6 Rechte und Pflichten der Lizenznehmer

Um eine Software in rechtskonformer Weise nutzen zu können, benötigt der Nutzer ein Nutzungsrecht (Lizenz) und die Software selbst. Entsprechend enthält ein Lizenzvertrag über Software zum einen die Auflagen und Bedingungen, unter denen das Nutzungsrecht an der Software auf den Lizenznehmer (Softwarenutzer) übertragen wird, und zum anderen Regelungen zur Verantwortlichkeit für die Software (z.B. Gewährleistung). Die Lizenz zur Nutzung einer Open-Source-Software gilt als erteilt, wenn und solange der Nutzer die Software im Einklang mit den Lizenzbedingungen einsetzt. Verstößt ein Lizenznehmer gegen die Lizenzbedingungen, kann er das Nutzungsrecht verlieren. Solche rechtlichen Vorgaben, bei deren Nichtbeachtung dem Adressaten ein rechtlicher Nachteil (z.B. Verlust des Nutzungsrechts) droht, werden Obliegenheiten genannt.

Als Lizenznehmer kann üblicherweise ein Mensch („natürliche Person“) oder eine juristische Person (etwa eine Gesellschaft in einer bestimmten Rechtsform) auftreten. Gängige Open-Source-Lizenzen ermöglichen oft beide Alternativen. Die Open-Source-Lizenz räumt dort beschriebene Rechte jedenfalls zunächst nur dem Lizenznehmer selbst ein – nicht anderen natürlichen oder juristischen Personen. Ausnahmen davon müssen sich aus dem Lizenztext hinreichend klar ergeben, wie etwa bei der Apache License Version 2.0, die als Lizenznehmer auch konzernangehörige Unternehmen des Empfängers der OSS definiert. Vielfach ergibt erst die Auslegung einer Open-Source-Lizenz, wer als Lizenznehmer zur Nutzung der Software berechtigt sein soll.

6.6.1 Mechanik der Rechtseinräumung

Bei proprietären Lizenzkonzepten werden die Nutzungsrechte an einer Software zwischen den Vertragspartnern weitergegeben. Entweder erhält ein Kunde eine Lizenz vom Urheber und räumt seinen eigenen Kunden Sublizenzen ein. Oder es werden beim Handel mit Software vordefinierte Lizenzen unverändert weiteregegeben, wobei dem Veräußerer keine eigenen Nutzungsrechte verbleiben.

Bei Open-Source-Software ist die Lizenzierung ganz überwiegend anders ausgestaltet. Hier leitet sich das Nutzungsrecht im Regelfall direkt und unmittelbar vom Urheber ab. § 10 der GPL 3 stellt dies ausdrücklich klar. Sind mehrere Urheber beteiligt, wie beispielsweise bei kollaborativen Softwareentwicklungsprojekten oder bei der Veröffentlichung von Modifikationen, führt dies dazu, dass die Rechte am Gesamtpaket aus unterschiedlichen Quellen stammen.

Kommentiert [KT16]: Kommentar Kriesel: Hier Grafik sinnvoll mit Abbildungen, wie die Lizenzbeziehungen in der Lieferkette von Software aussehen, an welcher Stelle Lizenzverträge abgeschlossen werden und an welcher Stelle nicht

Ein Nachteil des direkten Lizenzierungskonzeptes besteht in der komplexeren Rechtsdurchsetzung. Vorteil der Lizenzierung direkt vom Urheber ist allerdings, dass anders als bei der Sublizenzierung keine Mängel in der Rechtekette möglich sind, die zum Wegfall der Nutzungsrechte beim Endnutzer führen könnten. Selbst wenn ein Nutzer die Software unter Verstoß gegen die anwendbaren Lizenzbedingungen verbreitet (die Rechte dieses Nutzers erlöschen in diesem Fall), soll der Empfänger der Software dennoch nach manchen Lizenzen direkt vom Urheber lizenziert sein, soweit sich der Nutzer selbst an die Lizenzbedingungen hält. Dann ist in der Regel eine interne Nutzung nicht gefährdet.

Auf der anderen Seite muss der Nutzer daran interessiert sein, die Open-Source-Software lizenzkonform zu erhalten, um selbst keine Beeinträchtigung bei der Benutzung zu erleiden. Zumindest sollte sich der Nutzer vergewissern, ob die erhaltene Software unter einer Open-Source-Lizenz steht und wenn ja, unter welcher. Hierüber sollte sich der Empfänger eines Software-Pakets

beim Lieferanten informieren bzw. beim Lieferanten darauf hinwirken, dass diese Informationen zu OSS standardmäßig mitgeliefert werden.

Zwar setzen die Obliegenheiten aus Open-Source-Lizenzen teilweise an einer Weitergabe der OSS an. Aber eine solche Weitergabe kann bereits bei der Übertragung oder Bereitstellung der OSS durch ein Konzernunternehmen an ein anderes Konzernunternehmen vorliegen; denn Konzernunternehmen sind im Rechtssinn jeweils selbständige juristische Personen.

6.6.2. Rechte und Pflichten bei Nutzung von Open-Source-Software

Open-Source-Lizenzen räumen den Lizenznehmern zahlreiche Rechte ein und erlegen ihnen unterschiedliche Pflichten auf. Hierbei sind stets die konkreten Lizenzbedingungen zu beachten, da die Rechte und Pflichten je nach verwendeter Open-Source-Lizenz verschieden sein können. Es gibt jedoch einige Rechte und Pflichten, die den allermeisten Open-Source-Lizenzen gemeinsam sind; diese sollen im Folgenden dargestellt werden. An dieser Stelle muss zudem darauf hingewiesen werden, dass Informationsseiten, die bei der Auslegung von Lizenzen helfen, z.B. die FAQ der Free Software Foundation, ebenso wenig wie dieser Leitfaden abschließende Aussagen zu rechtlichen Problemstellungen treffen können. So kann es durchaus passieren, dass Miturheber von OSS bzw. Gerichte eine andere und ggf. strengere Interpretation vornehmen. Außerdem muss natürlich vorranging immer der jeweilige Lizenztext selbst zur Interpretation verwendet werden. Alle weiteren Informationsquellen können allenfalls ergänzend und unter bestimmten Voraussetzungen als Auslegungshilfe herangezogen werden.

Kommentiert [KT17]: Kommentar Herr Teichert: Hier wünsche ich mir eine juristisch korrekte Einleitung zum Thema wer ist Lizenznehmer bei einer Open Source Lizenz!

Bitte auf Lizenzformulierungen mit Arbeitstitel "Konzernklausel" wie in der Apache-2.0 eingehen, aber auch auf Unklarheiten in anderen Fällen hinweisen (falls diese besteht, ich bin kein Jurist).
Ein weiter Lizenznehmer Begriff ("Konzern") lößt m.E. weniger Lizenzpflichten aus, welche an einer Weitergabe der Software hängen.
Ein weiter Lizenznehmer Begriff ("Konzern") erfaßt m.E. ggf. mehr Patente und Patentstreitigkeiten, bei Patentfreigabeklauseln und Patentklageabwehrklauseln. Bitte
dringend einen Absatz „wer ist Lizenznehmer“ aufnehmen!

Kommentiert [KT18]: Anmerkung Kriesel zum Kommentar von Herrn Teichert: Entsprechende Ergänzungen finden sich nun unter 6.6, 6.6.1 und 6.6.2.

6.6.2.1 Rechte des Nutzers

Der Lizenznehmer ist in der Regel stets berechtigt, die Software

- zu nutzen, insbesondere auf seiner eigenen Hardware ablaufen zu lassen,
- zu vervielfältigen und zu verbreiten, insbesondere auch zum Download zugänglich zu machen,
- zu bearbeiten und weiter zu entwickeln, sowie diese bearbeiteten Versionen und Weiterentwicklungen zu verbreiten.

Die Weitergabe der OSS von einem Nutzer an einen weiteren Empfänger erfordert üblicherweise die Einhaltung dafür in der jeweiligen Open-Source-Lizenz vorgesehener Vorgaben, damit dem weiteren Empfänger ebenfalls die Nutzungsrechte nach dieser Open-Source-Lizenz verschafft werden. Um eine solche Weitergabe handelt es sich auch bei der Übertragung oder Bereitstellung der OSS durch ein Konzernunternehmen an ein anderes Konzernunternehmen, denn die Konzernunternehmen sind jeweils getrennte juristische Personen.

6.6.2.2 Pflichten des Nutzers

Voraussetzung für die Erlangung der oben (Ziffer 6.2.1.1) aufgeführten Rechte ist die Befolgung der in den Open-Source-Lizenzen aufgeführten Pflichten. Diese sind üblicherweise nur bei einer Weitergabe an Dritte – einschließlich verbundener Unternehmen – einschlägig und werden daher auch "Vertriebspflichten" genannt.

Bei der Verbreitung, der Veröffentlichung oder der Weitergabe sowohl unveränderter als auch veränderter Open-Source-Software sind durch den Nutzer im Regelfall folgende Pflichten zu beachten:

- Mitlieferung des konkreten Lizenztextes und des konkreten Haftungsausschlusses wie im Softwarepaket vorgegeben
- keine Veränderung von bestehenden Urhebervermerken und Haftungsausschlüssen, bzw. Pflicht zur Anbringung derselben an Vervielfältigungen

- keine Erhebung von Lizenzgebühren für die Verschaffung von Nutzungsrechten.

Der Lizenztext kann als Datei auf einem Datenträger oder in gedruckter Form mitgeliefert werden. Ein Verweis auf eine Webseite mit dem Lizenztext ist jedoch in der Regel nicht ausreichend. Zudem ist darauf zu achten, dass der identische Wortlaut des Lizenztextes für das jeweilige Softwarepaket verwendet wird. Es ist besonders darauf zu achten, dass stets die korrekte Version der Lizenz, unter der die jeweils verwendete Version der Open-Source-Komponente steht, mitgeliefert wird, nicht nur eine Standardvorlage.

Einige verbreitete Open-Source-Lizenzen sehen ebenfalls vor, dass der Source Code zugänglich zu machen ist. Dies kann je nach anwendbarer Lizenz auf verschiedene Weisen erfolgen, z.B. durch Mitlieferung, durch Abgabe eines Angebotes zur Lieferung des Source Codes an jeden Dritten, durch Übersendung eines Datenträgers mit dem Source Code oder durch Bereitstellung zum Download im Internet. Zwar besteht die Pflicht zur Zugänglichmachung des Source Codes nicht bei allen Open-Source- Lizenzen, es kann jedoch ratsam sein, stets den gesamten Source Code der Open-Source-Komponente mitzuliefern, da damit gleichsam der Pflicht zur Mitlieferung von Lizenztexten, Urhebervermerken, Haftungsausschlüssen etc. Genüge getan ist. Häufig wird auch noch verlangt, dass Änderungen am Quelltext dokumentiert und gekennzeichnet werden müssen, damit später eine Zuordnung zum jeweiligen Urheber möglich ist.

Neben den oben aufgeführten allgemeinen Pflichten des Nutzers können sich aus der jeweils anwendbaren Lizenz weitere spezielle Pflichten ergeben, die als Voraussetzung für die Geltung der durch die Lizenz eingeräumten Nutzungsrechte stets zu beachten sind.

Es liegt schließlich im Eigeneinteresse des Nutzers, sich seines Nutzungsrechts zu vergewissern und die Nutzungsberechtigung zu dokumentieren. Hierfür empfiehlt sich die Einrichtung eines Lizenzmanagements (vgl. hierzu näher die Ausführungen unter Ziffer 7.3.1).

6.6.3 Darstellung einiger lizenzspezifischer Rechte und Pflichten

6.6.3.1 BSD-Lizenzen

Bei der Gruppe der BSD-Lizenzen handelt es sich um eine der ältesten und dennoch bis zum heutigen Tage sehr weit verbreiteten Open-Source-Lizenzfamilie. "BSD" steht hierbei für Berkeley Software Distribution und bezieht sich auf ein unixartiges Betriebssystem, welches an der University of California, Berkeley entwickelt wurde.

Die BSD-Lizenzen zeichnen sich dadurch aus, dass sie permissiv, also nicht mit einem Copyleft-Effekt behaftet sind, d.h. es besteht insbesondere keine Pflicht, Modifikationen zu veröffentlichen oder ebenfalls unter die BSD-Lizenz zu stellen. Sie wird in Form eines Templates verbreitet, das auch von anderen Urhebern (ggf. unter abweichender Bezeichnung) verwendet wird.

Bei der aktuell am häufigsten verwendeten Version der BSD-Lizenzen handelt es sich um die 3-clause BSD-Lizenz (die auch unter der Bezeichnung "neue" oder "modifizierte" BSD-Lizenz bekannt ist). Im Unterschied zur originalen 4-clause BSD-Lizenz enthält diese Fassung keine sog. Werbeklausel mehr, nach der in allen Werbematerialen auf die Verwendung der so lizenzierten Software hingewiesen werden musste. Aufgrund des vermehrten Einsatzes von Open-Source-Software erwies sich die Umsetzung der Werbeklausel als wenig praktikabel, zudem verhinderte sie die Kompatibilität zur GNU GPL. Die University of California, Berkeley, hat deshalb bereits 1999 erklärt, dass sie für den von ihr unter der originalen BSD-Lizenz veröffentlichen Code auf die Werbeklausel verzichtet. Dies bedeutet aber natürlich nicht, dass die Werbeklausel allgemein unbeachtlich ist, da darüber nur der jeweilige Urheber verfügen kann.

Die 3-clause BSD-Lizenz räumt folgende Rechte ein:

- Nutzung, Vervielfältigung und Verbreitung des Codes in unveränderter Form als Source Code oder in Binärform

- Nutzung, Vervielfältigung und Verbreitung des Codes in veränderter Form als Source Code oder in Binärform
- Verwendung im Rahmen von kommerzieller und nicht-kommerzieller Software.

Dies knüpft die 3-clause BSD-Lizenz an folgende Pflichten:

Die in der 3-clause BSD-Lizenz enthaltenen Pflichten können durch die Einhaltung der in Ziffer 6.2.1.2 aufgeführten Pflichten erfüllt werden.

Neben der 3-clause BSD ist heute auch eine weiter vereinfachte Version als 2-clause BSD verbreitet. Bei dieser Lizenzvariante wird abweichend von der 3-clause BSD keine Regelung hinsichtlich einer Namensnennung von (Mit-)Urhebern getroffen.

Den 2- und 3-clause-BSD Lizenzen ist die sog. MIT-Lizenz ähnlich, die auch als „X11“-Lizenz bekannt ist. Die in der MIT-Lizenz enthaltenen Pflichten können ebenfalls durch die Einhaltung der in Ziffer 6.2.1.2 aufgeführten Pflichten erfüllt werden.

6.6.3.2 Apache Software License 2.0

Die Version 2.0 der Apache Software License (ASL 2.0) wurde im Jahr 2004 veröffentlicht und stellt eine der am weitesten verbreiteten Open-Source-Lizenzen dar. Insbesondere stehen die beliebten Open-Source-Programme der Apache Software Foundation unter dieser Lizenz (z.B. der Apache HTTP-Server oder Apache Ant). Die beiden Vorgängerversionen 1.0 und 1.1 der Lizenz kommen heutzutage deutlich seltener vor, weshalb sich diese Betrachtung auch nur auf die aktuelle Version bezieht.

Bei der ASL 2.0 handelt es sich um eine permissive Lizenz.

Die ASL 2.0 gewährt folgende Rechte:

- (Weiter-)Verbreitungsrecht, Recht zur öffentlichen Vorführung, zur Erstellung von Bearbeitungen und Unterlizenzierungsrecht des nach ASL 2.0 lizenzierten Werks oder eines davon abgeleiteten Werks in Source Code oder Binärform
- Änderungen des nach ASL 2.0 lizenzierten Codes oder abgeleitete Werke können unter eine andere Lizenz gestellt werden, solange die Verwendung und der Weitervertrieb des Werks im Übrigen die Bedingungen der ASL 2.0 erfüllt
- Einräumung einer eingeschränkten Patentlizenz.

Damit verbunden ist die Einhaltung der in Ziffer 6.2.1.2 aufgeführten Pflichten.

Als Besonderheit besteht beim Weitervertrieb von abgeleiteten Werken die Pflicht, den Inhalt der "NOTICE"-Datei (sofern sie in einem betreffenden Paket vorhanden ist) ebenfalls weiterzugeben, solange das abgeleitete Werk Bestandteile verwendet, auf die sich der Inhalt der "NOTICE"-Datei bezieht. Dabei kann gewählt werden, ob ebenfalls einfach die "NOTICE"-Datei verwendet und mitgeliefert wird, ob die Pflicht mittels einer Anzeige, mittels Angabe im Source Code oder in der Dokumentation (falls mitgeliefert) oder mittels Angabe an sonstiger üblicher Stelle erfüllt wird.

Sonstige Hinweise:

Werden Modifikationen des unter ASL 2.0 stehenden Werkes bewusst an den Urheber des Originalwerks übermittelt (hierfür reicht z.B. bereits die Diskussion des geänderten Codes per E-Mail), dann kann der Urheber des Originalwerks diese Modifikationen zu den Bedingungen des ASL 2.0 in das Werk integrieren. Dies kann aber einfach verhindert werden, indem in der Kommunikation klargestellt wird, dass es sich nicht um einen Projektbeitrag handeln soll.

Weiter ist wichtig zu wissen, dass man als Beitragender allen Empfängern des unter ASL 2.0 stehenden Werks eine Patentlizenz in dem Umfang einräumt, in dem das für die spätere Nutzung des Werks unter ASL 2.0 erforderlich ist. Damit bringt der Beitragende zum Ausdruck, dass er gegen einen Nutzer des Werkes nicht mit Ansprüchen aus dem Patentrecht vorgehen wird. Gleichzeitig enthält die Lizenz eine Patent-Vergeltungsklausel die besagt, dass man als Verwender eines unter

ASL 2.0 stehenden Werks alle gemäß der ASL 2.0 eingeräumten Patentnutzungsrechte verliert, sobald man gerichtlich geltend macht, dass das Werk oder ein Beitrag hierzu eine Patentverletzung darstellt. Diese Mechanismen sind nicht nur auf das Empfängerunternehmen selbst beschränkt, sondern erstrecken sich nach dem Wortlaut der ASL 2.0 auf alle Unternehmen des Konzerns, dem ein Empfängerunternehmen angehört.

Kommentiert [KT19]: Kommentar Herr Teichert: Bin kein Jurist, aber ich denke eine Diskussion des Arbeitsbegriffs „Konzernklausel" ist hier, ggf. mit Verweis auf einen neuen Absatz unter 6.2, unverzichtbar.
Das Verschweigen, dass es beim Patentnutzungsfreigeben um den ganzen Konzern geht und auch Klagen anderer Konzerngesellschaften hier relevant sind, könnte m.E. zu Schadensersatzansprüchen führen.
Das möchte ich vermeiden.

6.6.3.3 GNU General Public License (GPL)

Bei der GNU General Public License bzw. deren Variante für Bibliotheken, der sog. GNU Lesser General Public License, handelt es sich um die Ursprungsform einer Copyleft-Lizenz. Die GPL stellt die mit Abstand am häufigsten eingesetzte Open-Source-Lizenz dar, wobei Version 2 immer noch weiter verbreitet ist als Version 3, die Mitte 2007 erschien. Die Verbreitung der GPL hat dazu geführt, dass es relativ umfangreiche Literatur gibt[49], die sich mit den Rechten und Pflichten aus dieser Lizenz beschäftigt. Aus diesem Grund hält sich dieser Beitrag hinsichtlich der allgemeinen Gesichtspunkte der GPL bewusst zurück und beschränkt sich auf seltener angesprochene, jedoch gleichsam wichtige Aspekte.

a) Allgemeines zur GPL

Die GPL versucht, mittels des Copyleft-Effekts sicherzustellen, dass GPL Software auch bei Veränderungen oder bei Einbindung in andere Software (insgesamt) der GPL unterstellt bleibt. Dies wird insbesondere durch den "viralen" Effekt der GPL erreicht: Die GPL bestimmt, dass alle Änderungen, Bearbeitungen oder abgeleiteten Werke ebenfalls unter die GPL zu stellen sind. So führt beispielsweise das Linken von GPL-Code zu proprietärem Code dazu, dass der eigentlich proprietäre Code ebenfalls unter GPL lizenziert werden und in der Folge auch als Quelltext offengelegt werden muss. Dabei spielt es nach der sehr weiten Auffassung der Free Software Foundation keine Rolle, ob nur statisch oder dynamisch gelinkt wird.

b) Einschränkungen des Copyleft-Effekts

Um den Einsatz der GPL praxistauglich zu machen, wurde eine große Anzahl an Ausnahmeregelungen geschaffen, die insbesondere beim Einsatz von Tools aus Entwicklungsumgebungen zum Tragen kommen (z.B. die GCC-Linking Exception, die Classpath Exception, die Bison Exception, die Autoconf Exception etc.). Da viele der GNU Tools zur Softwareentwicklung Teile von sich selbst in den von ihnen generierten Output einkopieren bzw. integrieren, wäre dieser gemäß des starken Copylefts der GPL ebenfalls automatisch immer der GPL zu unterstellen. Die vorgenannten Ausnahmen verhindern deshalb innerhalb jeweils genau festgelegter Parameter das Eintreten des Copyleft-Effekts und eröffnen den Anwendern damit die Möglichkeit, die Lizenz für die Ergebnisse bzw. die erstellten Dateien selbst festzulegen. Hier gilt jedoch: Das Vorliegen der Voraussetzungen dieser Ausnahmeregelungen muss in jedem Einzelfall genau untersucht werden.

c) Interne Nutzung ohne Veröffentlichung/Weitergabe an Dritte

Die Pflichten der GPL sind nur dann zu erfüllen, wenn die Software weiterverbreitet wird. Eine rein interne Nutzung löst weder den Copyleft-Effekt aus, noch verpflichtet die GPL zur Offenlegung von im Rahmen privater oder interner Nutzung gemachten Änderungen. Wird die Software aber weitergegeben oder erhalten Dritte die Möglichkeit, sich Kopien anzufertigen, dann greifen die Pflichten der GPL wieder ein.

d) Häufige Fehler

Bei Verwendung von GPL-lizenzierter Software kommt es häufig zu vermeidbaren Fehlern, die aber insoweit kritisch sind, als sie einen Lizenzverstoß darstellen und deshalb nach der Logik der GPL zum Fortfall aller Lizenzrechte führen. Zu diesen Fehlern gehört es unter anderem, dass bei der

[49] Z.B. „Die GPL kommentiert und erklärt", Institut für Rechtsfragen der Freien und Open-Source-Software, http://www.oreilly.de/german/freebooks/gplger/; Jaeger/Metzger, Open-Source-Software, 3. Auflage 2011, Kapitel A. I - III

Verbreitung von GPL-lizenzierter Software der entsprechende Lizenztext nicht mitgeliefert wird. Eine weitere "beliebte" Fehlerquelle betrifft die Zugänglichmachung des Quelltexts, z.B. wenn nicht exakt der Quelltext mitgeliefert wird, der sich auf die im Objektcode vorliegende Software bezieht, oder wenn bei bestimmten Vertriebsformen nur auf den Quelltext verwiesen wird. Vielfach wird auch übersehen, dass es nicht ausreicht, nur den Quellcode als solchen bereit zu stellen, sondern dass auch die Skripte zur Steuerung von Kompilierung und Installation der Anwendung mitgeliefert werden müssen. Der Empfänger soll in die Lage versetzt werden, Änderungen oder Erweiterungen der Software durchführen zu können, ohne vorher größere technische Hürden überwinden zu müssen.

Konsequenz aus einem Verstoß gegen die Vorgaben der GPL 2 ist der automatische Verlust aller durch die Lizenz eingeräumter Nutzungsrechte für denjenigen, der den Verstoß begangen hat. Für den deutschen Rechtsraum wird von der vorherrschenden Rechtsmeinung angenommen, dass die GPL eine auflösend bedingte Rechtseinräumung nach § 158 Abs. 2 BGB enthält. Der Empfänger erhält die jeweiligen Nutzungsrechte nur unter der Bedingung, dass er alle Lizenzbedingungen einhält. Im Falle eines Verstoßes gegen eine Lizenzpflicht fallen alle eingeräumten Nutzungsrechte automatisch mit Wirkung für die Zukunft weg.[50] Empfänger, die eine Software unter GPL 2 von einem Verletzer erhalten haben, verlieren ihre Rechte aus der Lizenz nicht, wenn sie selbst alle Pflichten der GPL 2 einhalten.

e) Änderungen in Version 3 der GPL

Neben Änderungen und Klarstellungen bei den Begrifflichkeiten wurde in Version 3 der GPL u.a. in Ziffer 7 die Möglichkeit geschaffen, aus einem Katalog mit weiteren Einschränkungen zu wählen. Dies wäre in Version 2 der GPL aufgrund des Verbots weitergehender Beschränkungen nicht möglich gewesen. Neben dem Katalog an zulässigen weiteren Einschränkungen statuiert Ziffer 7 auch die Möglichkeit, Ausnahmen von der GPL V3 bzw. Befreiungen von deren Pflichten vorsehen kann. Eine weitere Änderung in Version 3 stellt die Aufnahme von ausdrücklichen und sehr umfangreichen Regelungen zur Patentlizenzierung dar. Zudem hat die Version 3 der GPL eine praxisnähere Lösung im Fall von Verstößen gegen die Lizenzpflichten aufgenommen. Abweichend zur Version 2 entfallen nicht mehr sofort alle Nutzungsrechte, sondern es gibt unter gewissen Umständen Möglichkeiten zur Heilung von Lizenzverletzungen innerhalb festgelegter Fristen.

6.6.3.4 LGPL

a) Allgemeines

Die LGPL bezieht sich auf Software-Bibliotheken und erlaubt unter bestimmten Voraussetzungen die Kombination der LGPL-Bibliothek mit Softwarebestandteilen, die unter einer anderen Lizenz stehen. In diesen Fällen wird die Wirkung des starken Copylefts der GPL eingeschränkt. Damit wird eine Nutzung solcher Komponenten in kommerzieller Software möglich und ein Übergreifen der GPL-Lizenzbedingungen auf die übrigen (insbesondere proprietären) Softwarekomponenten verhindert. Bei richtiger Verwendung von LGPL-Bibliotheken werden nur Änderungen an der Bibliothek selbst vom Copyleft erfasst, nicht aber das zugreifende Programm. Vielfach findet sich hier nur der Hinweis, dass auf eine dynamische Verlinkung zu achten sei, um dieses Ziel zu erreichen. Dies ist zwar zutreffend, stellt aber nur ein Kriterium für die lizenzgemäße Verwendung dar. Daneben sieht die LGPL 2.1 noch Regelungen zur Zurückentwicklung vor. Neben der allgemeinen Pflicht, auch den Lizenztext der LGPL 2.1 mitzuliefern und auf den Einsatz der LGPL hinzuweisen, bietet die LGPL 2.1 noch einen relativ komplexen Pflichtenkatalog an, aus dem eine Möglichkeit gewählt werden muss (u.a. Beifügen des kompletten maschinenlesbaren Quelltexts der LGPL-Bibliothek oder Beifügen eines schriftlichen Angebots, gültig für mindestens drei Jahre, wobei nur angemessene Herstellungskosten für den Datenträger berechnet werden dürfen). Auch bei der LGPL 2/2.1 führt ein Verstoß gegen eine der Lizenzpflichten zum sofortigen Verlust der Nutzungsrechte für denjenigen, der den Verstoß begangen hat (siehe Ziffer 6.2.2.3 d)).

b) Änderungen in Version 3 der GPL

---

[50] Jaeger / Metzger, Open-Source-Software, 3. Auflage 2011, Kapitel D.I. Rn. 152 ff.

Eine wesentliche Änderung in Version 3 der LGPL (die zunächst die GPL 3 als Ganzes einbezieht und dann Abweichungen definiert) besteht darin, dass die Bearbeitung des auf die LGPL-Bibliothek zugreifenden Programms nicht mehr erlaubt werden muss. Auch finden die für die GPL 3 geschaffenen Regelungen zur Patentlizenzierung und zum Digital Rights Management Anwendung.

6.6.4 Open-Source-Software beim Cloud Computing

Die Pflichten aus den OSS-Lizenzen greifen ein, wenn die Open-Source-Software an einen Nutzer überlassen wird, d.h. sie knüpfen an die Überlassung eines Vervielfältigungsstücks an. Diese kann durch Übergabe eines physischen Datenträgers oder im Wege des Downloads erfolgen. Beim Cloud Computing findet die Überlassung eines Vervielfältigungsstücks regelmäßig nicht mehr statt. Der Begriff Cloud Computing ist ein Überbegriff für die unterschiedlichsten Geschäftsmodelle, bei denen Nutzer Software und / oder Hardware über ein Netzwerk nutzen.[51] Wird ein Software-Programm im Rahmen des Cloud Computing nicht mehr auf den Nutzer übertragen, so ist fraglich, ob die Ermöglichung der Softwarenutzung über das Cloud-Netzwerk unter den in Open-Source-Lizenzen häufig verwendeten Begriff der "Verbreitung" (englisch "to distribute") fällt.

In einigen Rechtsordnungen wie dem U.S.-Recht, das aufgrund der Herkunft vieler Open-Source-Lizenzbedingungen für die Auslegung heranzuziehen sein wird, ist der Begriff der "Verbreitung" extrem weit. Darunter fallen auch die Vermietung im urheberrechtlichen Sinn sowie die öffentliche Zugänglichmachung.

Für das deutsche Urheberrecht dürfte man in Bezug auf das Recht der öffentlichen Zugänglichmachung (§ 69c Nr. 4 UrhG) im Wege der Auslegung zu einem identischen Ergebnis kommen. Als Argument hierfür könnte man anführen, dass viele Open-Source-Lizenzen von einer Zugänglichmachung der Software über das Internet ausgehen (vgl. z.B. für die GPL 2: "If distribution of executable or object code is made by offering access to copy from a designated place…"). Es gibt in dieser Frage allerdings bislang keine allgemein anerkannte Sichtweise und auch Auswirkungen verschiedener technischer Implementierungen sind noch nicht abschließend geklärt.

In den Lizenzen selbst finden sich nur teilweise Aussagen hierzu. So unterscheidet die GPL 3 zwischen den Nutzungshandlungen "convey" und "propagate". Während "convey" die Verbreitung oder die Bereitstellung von Kopien meint und Interaktionen über ein Netzwerk ohne Übergabe einer Kopie ausdrücklich ausschließt, umfasst der Begriff "propagate" alle sonstigen urheberrechtlich relevanten Sachverhalte. Obgleich beide Tätigkeiten von der Lizenzeinräumung umfasst sind, greifen die zusätzlichen Anforderungen der GPL 3 nur, wenn auch eine Überlassung der Software ("convey") vorliegt.

Kommentiert [KT20]: Kommentar Kriesel: Aufgrund des Kommentars von Herrn Teichert gestrichen.

Subsumiert man Cloud-Sachverhalte unter den Begriff der "Verbreitung", so bedeutet dies, dass die Anforderungen an eine solche Verbreitung (z.B. Verfügbarmachung des Source Codes) eingehalten werden müssen, soweit die Lizenzbedingungen nicht etwas anderes vorsehen.

Sieht man die Bereitstellung einer OSS im sogenannten „Software-as-a-Service“ Verfahren (SaaS) nicht als Überlassung an und treffen die einschlägigen Lizenzbedingungen insoweit keine Aussagen, so kann der Cloud-Anbieter seinen Kunden die Nutzung der OSS ermöglichen, ohne dabei die OSS-Lizenzbestimmungen für eine Weitergabe einhalten zu müssen. Das betrifft insbesondere die GPLv2 / LGPVv2.1, die für die klassische Verteilung von Software per Datenträger oder zumindest per Download entworfen wurden. Als Konsequenz kann ein Anbieter erweiterte oder verbesserte GPL-lizenzierte Komponenten in SaaS-Projekten benutzen, ohne den Quellcode der Projekte jemandem

[51] Einzelheiten zum Cloud Computing sind dem BITKOM-Leitfaden „Cloud Computing – Was Entscheider wissen müssen“ (http://www.bitkom.org/de/publikationen/38337_66148.aspx) zu entnehmen.

außerhalb des eigenen Unternehmens zugänglich machen zu müssen, da weder Wettbewerber noch Nutzer, die die Software selbst betreiben wollen, einen Rechtsanspruch darauf haben.

Um diese kritische Schwachstelle der GPL (sog. GPL-Loophole) zu schließen, schuf der Anbieter Affero die sogenannte GNU Affero General Public License (AGPL) mit ihrem Verweis auf die Bereitstellung über Netzwerke. Die AGPL führt im Ergebnis dazu, dass eine Bereitstellung der darunter lizensierten OSS im SaaS-Verfahren eine „Weitergabe“ darstellt und daher alle bei einer „normalen“ GPL für die Weitergabe geltenden Pflichten des Anbieters, insbesondere die Pflicht zur Offenlegung des Quellcodes, zu erfüllen sind. Für Anbieter, die Lösungen mit modifizierten OSS-Komponenten erstellt haben und diese im SaaS Verfahren anbieten wollen, ist es daher von kritischer Bedeutung, bei der Auswahl der eingesetzten OSS solche AGPL-Lizenzen auszuschließen.

6.7 Lizenzerfüllung

Der rechtskonforme Einsatz einer Open-Source-Software besteht in der Befolgung der für die Software jeweils geltenden Lizenzbedingungen. Dies setzt zunächst voraus, dass die Lizenzen komplett erfasst werden. Deren Anzahl ergibt sich nicht nur aus den vordergründig involvierten Open-Source-Paketen und den damit verbundenen Lizenzen. Zusätzlich können diese Pakete weitere Pakete beinhalten, die weitere Lizenzen nach sich ziehen usw. Die mögliche Kaskadierung macht deutlich, dass eine vollständige Erfassung auf manuelle Weise arbeitsaufwändig bis unmöglich sein kann. In der Regel werden daher Softwarewerkzeuge zur Erfassung verwendet.

Des Weiteren besteht die Aufgabe, eine rechtsverbindliche Interpretation einer Lizenz für deren Einhaltung zur Verfügung zu stellen. Ziel ist, für das Unternehmen die Rechtssicherheit für die Auslieferung der Software herzustellen. Soweit Interpretationen und Umsetzungsvorschläge für gängige Lizenzen bereits publiziert wurden, darf dies nicht zu dem Missverständnis führen, dass eine solche generelle Interpretation ausreichend ist. Vielmehr ist die Umsetzung von Lizenzbedingungen in jedem Einzelfall anhand der konkret vorliegenden Lizenz zu überprüfen.

Für eher wenig bekannte, individuelle Open-Source-Lizenzen, die beispielsweise nur von einzelnen Autoren verwendet werden, muss ebenfalls eine rechtlich fundierte Einschätzung der Konsequenzen der Lizenzbedingungen vorliegen. Diese müsste ggf. erstellt werden.

Allerdings lässt sich nicht jeder Einsatzfall rechtlich abschließend beurteilen, da sich Rechtsprechung und Rechtsauffassung laufend fortentwickeln. Es gibt im Bereich der Open-Source-Software wiederholt Beispiele unterschiedlicher Rechtsauffassung, die noch nicht durch Rechtsprechung entschieden wurden. Die juristische Arbeit liegt dann darin, die offene Rechtsfrage zu identifizieren und auf diese hinzuweisen, um somit bei einer Risikobeurteilung dem Management zur Hand zu gehen.

6.8 Folgen von Lizenzverletzungen

6.8.1 Rechtsfolgen aus dem Lizenzvertrag

Besonderheiten in vertraglicher Hinsicht ergeben sich insbesondere bei den sog. bei „Copyleft“-Lizenzen. Bei den meisten dieser Lizenzen entfällt das Nutzungsrecht beim Verstoß gegen eine der Lizenzbedingungen automatisch. Die lizenzkonforme Nutzung ist Bedingung für die Zustimmung des Urhebers zur Nutzung. Bei Verstößen gegen die Lizenzbedingungen erlischt die Zustimmung, und es ist häufig in diesen Lizenzen geregelt, dass sämtliche zuvor eingeräumten Nutzungsrechte entfallen. Rechtlich wird dies als „auflösende Bedingung“ bezeichnet. Einige OSS-Lizenzen sehen im Fall von Lizenzverstößen einen Zeitraum vor, in dem der Verstoß „geheilt“ werden kann, bevor solche schwerwiegende Rechtsfolgen eintreten. Wichtig ist hierbei, dass der Fortfall der Nutzungsrechte sich

üblicherweise nicht auf Kopien erstreckt, die ein Dritter vom Verletzer erhalten hat. Dies liegt in der Tatsache begründet, dass im Rahmen von OSS-Lizenzen üblicherweise eine Direktlizenzierung vom jeweiligen Urheber zum Empfänger des OSS-lizenzierten Codes erfolgt und keine klassische „Rechtekette“ existiert. Spätere Empfänger des Codes sind in diesem Fall nicht von der Beendigung der Nutzungsrechte des Verletzers betroffen. Viele OSS-Lizenzen, die dieses „scharfe Schwert“ des Rechtefortfalls als Druckmittel zur Einhaltung der Lizenzpflichten verwenden, stellen diesen Verbleib der Nutzungsrechte bei späteren Empfängern noch zusätzlich klar.

6.8.2 Rechtsfolgen aus dem Urheberrecht

Grundsätzlich setzen Ansprüche aus dem Urheberrechtsgesetz (UrhG) wegen Verletzung des Urheberrechts weder die absichtliche noch die bewusste Verletzung des Urheberrechts voraus. Teilweise ist aber ein Verschulden des Verletzers (zumindest Fahrlässigkeit) erforderlich. Die Ansprüche des Urhebers bestimmen sich nach den §§ 97 ff des Urheberrechtsgesetzes (UrhG). Sie sind im Folgenden kurz dargestellt.

6.8.2.1 Anspruch auf Unterlassung (§ 97 Abs. 1 UrhG)

Der Rechteinhaber hat gegen den Verletzer einen Anspruch auf Beseitigung und Unterlassung weiterer oder erstmalig drohender Verletzungen in Zukunft. Dieser Anspruch ist verschuldensunabhängig. Der Anspruch wird zunächst mit einer Abmahnung (§ 97a UrhG) geltend gemacht.

6.8.2.2 Anspruch auf Schadensersatz (§ 97 Abs. 2 UrhG)

Dem Rechteinhaber steht bei einer Urheberrechtsverletzung ferner ein verschuldensabhängiger Schadensersatzanspruch zu. Dabei ist nur der Schaden zu ersetzen, der dem Verletzten tatsächlich entstanden ist. Eine Genugtuung, Abschreckung oder Bestrafung ist hingegen nicht Gegenstand des Schadensersatzanspruchs. Der Verletzte kann die Schadenshöhe nach der Rechtsprechung auf drei verschiedene Arten berechnen, nämlich nach dem Verletzergewinn, dem konkreten Schaden oder der sog. Lizenzanalogie. Bei letzterer entspricht der Schaden einer angemessenen Lizenzgebühr.

Auch bei Open-Source-Software kann ein Schadensersatzanspruch auf Grund einer fiktiven Lizenzgebühr gefordert werden, obwohl die Software kostenlos ist. Denn Voraussetzung für die Unentgeltlichkeit ist die Einhaltung der Lizenzbedingungen. Werden diese verletzt, wird die fiktive Lizenzgebühr nach dem Marktpreis für eine unbeschränkte Lizenz bemessen. Es wird daher gefragt, was es den Verwender gekostet hätte, wenn er für sein (lizenzuntreues) Verhalten vorher eine Erlaubnis eingeholt hätte. Dies wird vom Gericht frei geschätzt.

6.8.2.3 Anspruch auf Vernichtung, Rückruf, Überlassung (§§ 69f, 98 UrhG)

Die Anspruchsgrundlagen der §§ 69f, 98 UrhG sehen einen verschuldensunabhängigen Vernichtungsanspruch vor. Der Verletzte kann die Vernichtung der Vervielfältigungsstücke und der Vervielfältigungsvorrichtungen, wenn diese der Vervielfältigung gedient haben, verlangen. § 98 Abs. 2 regelt den Anspruch auf Rückruf und auf Entfernung der Vervielfältigungsstücke aus den Vertriebswegen. Als Alternative zu dem Vernichtungsanspruch kann der Rechteinhaber als Folge einer Verletzung seines Schutzrechtes die Überlassung der Vervielfältigungsstücke gegen eine angemessene Vergütung verlangen (§ 98 Abs.3, § 69f Abs. 1 UrhG).

Dies kann bei Open-Source-Software insbesondere dann relevant werden, wenn sie in Hardware, z.B. einem Router „verbaut“ ist. Im Extremfall, wenn kein milderes Mittel (Löschung) gleich wirksam ist, müssen sogar die Endprodukte vernichtet werden!

6.8.2.4 Anspruch auf Auskunft und Einsicht (§§ 101, 101a UrhG)

Bei Lizenzverstößen kann der Urheber Auskunft nach § 101 UrhG verlangen. Praktisch wird der Urheber aber schon Nachweisschwierigkeiten haben, dass sein Quellcode ohne seine Zustimmung verwendet worden ist. Dafür bietet § 101a UrhG ihm schon dann ein Einsichtsrecht in den Quellcode des fraglichen Programmes, wenn eine Verletzung zwar nicht bewiesen, aber „hinreichend wahrscheinlich“ ist. Bei der Frage, ob ein solcher Anspruch besteht, ist zu berücksichtigen, dass einerseits die Einsichtnahme oft einziges sicheres Mittel zum Beweis der Verletzung ist und andererseits, dass der Quellcode oft ein ganz essentielles Geschäftsgeheimnis des Beschuldigten darstellt.

6.8.2.5 Strafbarkeit wegen unerlaubter Verwertung (§106 UrhG)

Der Tatbestand des § 106 Abs. 1 UrhG stellt das Vervielfältigen, das Verbreiten oder die öffentliche Wiedergabe urheberrechtlich geschützter Werke ohne Einwilligung des Rechteinhabers unter Strafe (Freiheitsstrafe bis zu 3 Jahren oder Geldstrafe). Zur Erfüllung des Tatbestandes ist Vorsatz erforderlich; bloßes fahrlässiges Handeln reicht nicht.

Die Staatsanwaltschaft muss jedoch keine öffentliche Klage erheben, sondern kann Geschädigte auf den praktisch wenig relevanten strafrechtlichen Privatklageweg verweisen, wenn kein öffentliches Interesse an einer Strafverfolgung besteht.

6.8.3 Sonstiges

Zum Abschluss soll noch auf einen weiteren „Fallstrick“ bei der Verwendung von Open-Source-Software aufmerksam gemacht werden: Lizenzen zur Nutzung eines Programmes erstrecken sich nicht auf andere gewerbliche Schutzrechte. Insbesondere das Markenrecht bleibt daher unangetastet.

Es sollte daher immer darauf geachtet werden, dass eigene Programme, die zulässigerweise auf einer Open-Source-Software basieren, nicht unter Verwendung des Namens der Open-Source-Software oder deren Symbolik angeboten werden. Können mögliche Abnehmer den Eindruck gewinnen, dass das angebotene Programm vom Hersteller der Open-Source-Software stammt, kann eine Herkunftstäuschung vorliegen. Dies kann zu marken- und wettbewerbsrechtlichen Konsequenzen führen.

6.9 Haftung, Gewährleistung und deren Ausschluss

Die meisten OSS-Lizenzen wurden vor dem Hintergrund des US-amerikanischen Rechts konzipiert. Aus Sicht eines deutschen Verwenders von OSS, aber auch eines Urhebers, der ein Werk unter eine solche OSS-Lizenz stellen möchte, kann zweifelhaft erscheinen, ob die hinter diesen Lizenzen stehenden Grundsätze des US-amerikanischen Rechts uneingeschränkt auch für das deutsche Recht Anwendung finden. Dies gilt vor allem für die sehr weit gehenden Haftungs- und Gewährleistungsausschlüsse und für die Vereinbarkeit mit dem Recht der Allgemeinen Geschäftsbedingungen (AGB).

6.9.1 AGB-Recht

OSS-Lizenzen lassen sich als Allgemeine Geschäftsbedingungen (AGB) i.S.d. §§ 305 ff. BGB einordnen. Die Texte der OSS-Lizenzen sind für eine Vielzahl von Verwendungen vorformuliert und werden durchweg vom Urheber bzw. Rechtsinhaber vorgegeben, ohne dass sie einzeln verhandelt

werden würden. Dies gilt auch, wenn eine Software „dual licensed“ oder gar unter einer Vielzahl von Lizenzen angeboten wird - jeder dieser Lizenztexte ist für sich ja wiederum vorformuliert.[52]

AGB unterliegen der Wirksamkeitskontrolle der §§ 305 ff. BGB. Insoweit dürfen sie den anderen Vertragspartner nicht gegen das Gebot von Treu und Glauben unangemessen benachteiligen, da sie ansonsten unwirksam sind (§ 307 Abs. 1 Satz 1 BGB). Eine unangemessene Benachteiligung besteht nach § 307 Abs. 2 Nr. 1 BGB insbesondere dann, wenn AGB von wesentlichen Grundgedanken einer einschlägigen gesetzlichen Regelung abweichen und diese Abweichung mit der Intention des Gesetzgebers nicht vereinbar ist.

Darüber hinaus enthalten insbesondere §§ 308 und 309 BGB zahlreiche Tatbestände, die zur Unwirksamkeit von AGB führen können. Zwar gelten diese wegen § 310 Abs. 1 BGB unmittelbar nur zugunsten von Verbrauchern. In der Praxis fließen jedoch die in den §§ 308 und 309 BGB enthaltenen Wertungen des Gesetzgebers in die durch Gerichte vorgenommene Bestimmung einer „unangemessenen“ Benachteiligung i.S.d. § 307 Abs. 1 Satz 1 BGB mit ein. Und § 307 BGB findet auch für Verträge mit Unternehmen Anwendung. Sind OSS-Lizenzen als AGB einzuordnen, müssen sie sich dementsprechend bei Geltung deutschen Rechts auch am deutschen Standard für AGB messen lassen.

Dies spielt aber nur dann eine Rolle, wenn die AGB bzw. die OSS-Lizenzen wirksam in den Vertrag einbezogen wurden (§ 305 Abs. 2 BGB). Dazu ist erforderlich, dass der Vertragspartner mindestens die Möglichkeit einer zumutbaren Kenntnisnahme der AGB „bei Vertragsschluss“ erhält und sein „Einverständnis“ mit den AGB signalisiert. Insofern gelten für OSS-Lizenzen keine anderen Anforderungen als bei jedem anderen Softwareüberlassungsvertrag auch. Kann der Erwerber einer Software vor Abschluss des Überlassungsvertrags keine zumutbare Kenntnis von den AGB, hier also den OSS-Lizenzbestimmungen, nehmen, sind diese möglicherweise schon gar nicht Vertragsbestandteil geworden. In der Praxis sollte daher ein Unternehmen, welches OSS in seinen Produkten einsetzt, darauf achten, dass der Kunde die Möglichkeit hat, vor dem Abschluss des Lizenzvertrages Kenntnis von den OSS-Lizenzen zu nehmen, so wie dies auch für die Lizenzbedingungen für die proprietären Komponenten des Produkts gilt.

Unproblematisch zu bejahen ist die Möglichkeit zur Kenntnisnahme, wenn der Erwerber einer OSS oder eines Produktes mit Anteilen von OSS die Texte der OSS-Lizenzen vor dem Erwerb ausgehändigt bekommt. Praktisch weitaus häufiger dürfte jedoch der Fall sein, dass die OSS als einzelne Komponente oder als Teil eines Produktes per Download über das Internet bezogen wird. Dann stellt sich die Frage, ob und wann eine zumutbare Möglichkeit der Kenntnisnahme gegeben ist. Manchmal sind die Lizenzbestimmungen zu einzelnen OSS-Komponenten an der Stelle hinterlegt, von der der Download erfolgt. Dies ist aber nicht zwingend der Fall. Und schon bei etwas komplexeren Paketen von OSS mit einer Vielzahl von unterschiedlich lizenzierten Komponenten (z.B. einer Entwicklungsumgebung) wird zweifelhaft, ob es noch „zumutbar“ ist, hunderte von einzelnen OSS-Lizenzbestimmungen zusammenzusuchen und zur Kenntnis zu nehmen. Eine praktikable Möglichkeit, dem Rechnung zu tragen, ist es, eine Zusammenstellung der relevanten OSS-Lizenzen in einer einzelnen Datei vor oder mit dem Download des Pakets anzuzeigen oder zumindest separat zum Download anzubieten.

Rechtlich bedenklich sind die in der Praxis immer noch häufig zu beobachtenden sogenannten „Shrink-Wrap“ oder „Click-Wrap“ Lizenzen. Im Fall einer „Shrink-Wrap“ Lizenz erfährt der Kunde erst nach dem Öffnen der Verpackung, dass die - innenliegenden - Lizenzbestimmungen einschließlich der OSS-Lizenzen gelten sollen. Die moderne Form nennt sich „Click-Wrap“-Lizenz. Hierbei werden die (OSS-)Lizenzbestimmungen erst nach dem Herunterladen bei der Installation oder beim ersten Programmstart angezeigt, und der Anwender muss diese akzeptieren, um das Programm überhaupt installieren bzw. nutzen zu können. Nahezu einhellig sind deutsche Juristen der Auffassung, dass

[52] Dementsprechend hat auch das LG München die GPLv2 mit Urteil vom 19.5.2004 (Az. 21 O 6123/04) zwanglos und ohne größere Begründung als AGB eingestuft. Die Tatsache, dass die GPL in Englisch abgefasst sei, wurde vom Gericht als unproblematisch erachtet, weil die englische Sprache in der Computerindustrie die gängige Fachsprache sei, zumindest im Geschäftsverkehr zwischen Unternehmern.

solche „Shrink-Wrap- / Click-Wrap“-Lizenzen ohne hinreichende Hinweise auf den entsprechenden Lizenzvertragstext vor dem Vertragsschluss nicht wirksam in den Vertrag einbezogen werden können, da der Kunde erst nach dem Vertragsschluss (dem Kauf des Produkts im Laden oder dem kostenpflichtigen Download) auf die AGB hingewiesen wird.

Eine aufgrund der AGB-Vorschriften unwirksame Einbeziehung von Lizenzbedingungen in den Überlassungsvertrag für OSS bedeutet jedoch nicht, dass jegliches Nutzungsrecht entfällt. Ein Nutzungsverbot wäre nicht im Sinne des ursprünglichen Lizenzgebers: er wollte seine Software ja gerade unter den Bedingungen der Lizenz freigeben – auch wenn das wegen des deutschen AGB-Rechts nicht geglückt ist. Einen Ausweg bietet § 69d UrhG. Die Vorschrift ordnet an, dass der rechtmäßige Erwerber einer Software diese stets im bestimmungsgemäßen Umfang nutzen darf.[53] Damit dürfte der Nutzer die Software wenigstens einmal auf einem Rechner ‚installieren‘. Eine weitergehende Nutzung wie z.B. die Bearbeitung der Software erlaubt die Vorschrift aber nicht. Solange sich der Nutzer an die Lizenzvorgaben hält, obwohl sie aus AGB-Sicht nicht wirksam sind, ist es äußerst unwahrscheinlich, dass sich der ursprünglichen Lizenzgeber auf die Unwirksamkeit der Lizenzbedingungen nach deutschem Recht beruft. So stellt z.B. die GPLv2 klar: „*…Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.*“[54]

Teilweise wird auch vertreten, die OSS-Lizenzen seien ohnehin nur als Rechteeinräumungen zu dem Zweck anzusehen, dem Nutzer über den gesetzlichen Mindestumfang hinaus zusätzliche Rechte zu verschaffen, wie das Recht zur Bearbeitung und Weiterverbreitung der OSS.[55] Auch wenn man sich dieser Auffassung nicht anschließt, bleibt als juristische „Krücke“ die Nutzungsmöglichkeit nach § 69d UrhG.

6.9.2 Wirksamkeit von Gewährleistungsregelungen in OSS-Lizenzen

Sind OSS Lizenzen als AGB einzustufen, stellt sich weiterhin die Frage nach der Wirksamkeit der darin enthaltenen Gewährleistungsausschlüsse. Regelmäßig sind in OSS-Lizenzen weitgehende Gewährleistungsausschlüsse enthalten. So enthält z.B. die sogenannte „BSD“-Lizenz sinngemäß folgende Passage: „Diese Software wird von den Copyright-Inhabern bereitgestellt, so wie sie ist, und jegliche ausdrückliche oder implizite Gewährleistung, einschließlich, aber nicht abschließend, der impliziten Gewährleistung für die Verwendbarkeit und Geeignetheit für einen bestimmten Zweck, sind ausgeschlossen.“ Andere OSS-Lizenzen enthalten ähnlich weitgehende Regelungen, allesamt stark geprägt vom US-amerikanischen Rechtssystem. Praktisch relevant ist diese Frage vor allem für deutsche Anbieter, die OSS erstellen oder auch nur weitergeben, da es häufig kaum wirtschaftlich sein wird, vermeintliche Gewährleistungsrechte im Ausland durchzusetzen.

Zwar sind im deutschen Recht Gewährleistungspflichten bei der dauerhaften Softwareüberlassung individualvertraglich weitgehend ausschließbar, ausgenommen sind nur Fälle von Arglist (vgl. §§ 444, 639 BGB). Da aber OSS-Lizenzen als AGB zu qualifizieren sind, gilt hierfür der weitaus strengere Maßstab der §§ 305ff. BGB. Das führt im Ergebnis dazu, dass die vorgenannten Gewährleistungsausschlüsse nach deutschem Recht regelmäßig als unwirksam anzusehen sind.

Für einen deutschen Anbieter von OSS mag der Gedanke nahe liegen, die unwirksame Regelung einfach zu entfernen. Allerdings verpflichten fast alle OSS-Lizenzen dazu, die darin enthaltenen Gewährleistungsausschlüsse nicht zu entfernen. Nur dann soll man in den Genuss der Bearbeitungs-

---

[53] Die Regelung basiert auf der EU Computersoftwarerichtlinie (Richtlinie 91/250/EWG des Rates vom 14. Mai 1991), insofern gilt in den übrigen EU Ländern nichts fundamental anderes

[54] vgl. „inoffizielle“ Übersetzung der GPLv2, http://www.gnu.de/documents/gpl-2.0.de.html, dort § 4 Satz 3

[55] vgl. ifrOSS, Kommentar zur GNU General Public License, 2005, abzurufen unter http://www.ifross.org/ifross_html/gpl-seite.html

und Verbreitungsrechte kommen. Mit Ausnahme der wenigen Fälle, bei denen der Anbieter die OSS unter einer Lizenz beliebigen Inhalts und damit auch unter einer modifizierten OSS oder einer proprietären Lizenz weitergeben darf, muss sich daher jeder Anbieter, der OSS bearbeiten und / oder verbreiten möchte, fragen, welche Gewährleistungsregelungen denn gelten.

Wird OSS als integraler Bestandteil eines kommerziellen Produkts verbreitet, gilt die Gewährleistungsverpflichtung des Veräußerers für den proprietären Anteil auch für den OSS-Anteil; denn das BGB unterscheidet nicht danach, ob die Software selbst erstellt wurde oder fremde Komponenten, seien sie zugekauft oder eben OSS, enthält.

Wird hingegen die OSS als solche nicht verkauft, sondern unentgeltlich überlassen - etwa in der Form, wie sie jemand auch von der Webseite des Anbieters, des ursprünglichen Erstellers oder von einem Dritten erhalten könnte - und wird somit die OSS quasi „verschenkt", soll nach überwiegender Auffassung[56] das Schenkungsrecht (§§ 516ff. BGB) Anwendung finden. Bei einer verschenkten Software ist die Haftung für Sach- und Rechtsmängel von Gesetzes wegen auf Fälle von arglistig verschwiegenen Mängeln beschränkt (§§ 523 Abs. 1, 524 Abs. 1 BGB).

Dies bedeutet nicht, dass Hersteller von Softwareprodukten mit OSS-Anteilen sicher sein können, keinen Gewährleistungsansprüchen wegen der OSS ausgesetzt zu sein. Ein „arglistiges Verschweigen" von Mängeln kann z.B. schon vorliegen, wenn der Hersteller von OSS, die in einem als „sicher" beworbenen Produkt eingesetzt wird, positiv von einer für diese OSS relevanten Sicherheitslücke weiß, die die Sicherheit des Produkts kompromittiert, und trotzdem eine verfügbare aktuellere Version der OSS nicht eingesetzt wird. Fälle wie „Heartbleed" bei der sehr verbreiteten OSS „OpenSSL" haben gezeigt, dass dies keine theoretischen Risiken sind.

6.9.3 Wirksamkeit von Haftungsregelungen in OSS-Lizenzen

Ebenso wie die oben dargestellten Gewährleistungsausschlüsse enthalten fast alle OSS-Lizenzen einen Haftungsausschluss. So regelt etwa die GPLv2: „*…In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen….*".[57] Ein solcher pauschaler Ausschluss der Haftung ohne Unterscheidung von Vorsatz und (grober) Fahrlässigkeit oder auch ohne Differenzierung nach der gesetzlichen Grundlage (z.B. Produkthaftung) ist mit dem AGB-Recht nicht vereinbar und daher unwirksam.

Gleiches gilt, wie auch bei der Gewährleistung, für Klauseln, die die Haftung „*soweit gesetzlich zulässig*" ausschließen sollen. Während das „Herkunftsrecht" der meisten OSS-Lizenzen, das US-Recht, solche Klauseln zuzulassen scheint, sind solche Regelungen nach deutschem Recht durchweg unwirksam.

Anstelle der unwirksamen Haftungsbeschränkungen in den OSS-Lizenzen gelten daher die gesetzlichen Bestimmungen, die bei Annahme einer Schenkung eine Haftung nur im Umfang von § 521 BGB vorsehen, die auf Vorsatz und grobe Fahrlässigkeit beschränkt ist. Aus Sicht eines deutschen Anbieters muss dies nicht notwendigerweise ein Nachteil sein, da dies im Ergebnis durchaus über das hinausgeht, was in AGB vereinbar wäre[58]. Vertraglichen Beschränkungen ohnehin nicht zugänglich sind auch die zwingenden Haftungsvorschriften des Produkthaftungsgesetzes.

Allerdings besteht für deutsche Anbieter ein anderes Dilemma: Der vollständige Haftungs- und übrigens auch Gewährleistungsausschluss in den OSS-Lizenzen ist zwar gegenüber den Kunden des

---

[56] vgl. Jaeger / Metzger, Open Source Software, 3. Aufl. 2011, Rz. 205ff., insbes. Rz 217, mit vielen Nachweisen.
[57] vgl. „inoffizielle" Übersetzung der GPLv2, http://www.gnu.de/documents/gpl-2.0.de.html, dort § 12
[58] vgl. das „Kardinalpflichten"-Urteil des BGH vom 20.7.2005 (Az. VIII ZR 121/04).

Anbieters regelmäßig unwirksam, kann aber nach ausländischen Rechtsordnungen zumindest teilweise doch wirksam sein. Der deutsche Anbieter läuft daher Gefahr, seinen Kunden gegenüber zumindest im Umfang des Schenkungsrechts einstehen zu müssen, aber keinerlei Rückgriff beim Anbieter nehmen zu können, der aufgrund einer weitergehenden und nach ausländischem Recht wirksamen Klausel solche Ansprüche ausgeschlossen hat.

6.9.4 Außervertragliche Haftung und Mitverschulden

Wie oben ausgeführt (Ziffer 6.7.2), lässt sich die dauerhafte Überlassung einer OSS juristisch als Schenkung verstehen, wenn sie unentgeltlich erfolgt. Infolgedessen liegt für die Haftung eine Anwendung des § 521 BGB auch für das Deliktsrecht nahe.[59] Eine solche Haftungsprivilegierung würde jedoch nur zwischen den Vertragsparteien Anwendung finden. Denn eine Schenkung als schuldrechtlicher Vertrag kann nur Wirkungen zwischen den beteiligten Parteien entfalten.

Wird, verursacht durch Software, ein Dritter verletzt, kann § 521 BGB nicht zur Bestimmung des Haftungsmaßstabes herangezogen werden. Damit muss im Rahmen des § 823 BGB auf die Verkehrspflichten abgestellt werden. Funktion der Verkehrspflichten ist es, den Tatbestand der widerrechtlichen fahrlässigen Verletzung näher zu klären.

6.10 Rechtliche Fragen der Entwicklung von Open-Source-Software

6.10.1 Verträge für Erstellung und Änderung von Open-Source-Software

Bei der Erstellung und Änderung von OSS gelten keine grundlegend anderen Regelungen als für andere Software. Wird eine bestehende OSS bearbeitet, kann beim Bearbeiter der OSS ein Bearbeiterurheberrecht entstehen (vgl. oben Ziffer 6.5.2). Es steht jedermann frei, eigene Software als OSS zu veröffentlichen, soweit die dafür notwendigen Rechte vorhanden sind.

In der Praxis lassen sich drei Grundfälle unterscheiden, auch wenn weitere Konstellationen anzutreffen sind:

- **Fall 1:** Ein Entwickler möchte eigene Software als OSS veröffentlichen;
- **Fall 2:** Ein Entwickler nimmt an einer bestehenden OSS Änderungen vor;
- **Fall 3:** Ein Entwickler nimmt an einer bestehenden OSS Änderungen vor, möchte aber erreichen, dass die Änderung möglichst vielen zukünftigen Anwendern der OSS zugutekommt, auch wenn die zugehörige OSS-Lizenz eine Veröffentlichung nicht zwingend fordert.

Im **Fall 1** gilt, dass jeder, der über die dafür notwendigen Rechte verfügt, seine Software unter jeder beliebigen Lizenz veröffentlichen kann, sei es eine proprietäre oder eine OSS-Lizenz. Die Rechteinhaberschaft kann bei Projekten einzelner Personen oder bei eng miteinander verbundenen Gruppen relativ leicht sichergestellt werden. Ist eine Software das Ergebnis der Zusammenarbeit einer größeren Gruppe, oder will ein Unternehmen eine Software als OSS veröffentlichen, ist die Aufgabe ungleich komplexer. Dies gilt insbesondere dann, wenn es sich um größere Softwarepakete handelt und / oder Bestandteile enthalten sind, die (auch) durch externe Dritte erstellt wurden. Es empfiehlt sich, in diesen Fällen den Code genau zu analysieren und eine umfassende Rechteklärung vorzunehmen. Bei durch Dritte entwickelten Komponenten sollten die Entwicklungsverträge daraufhin geprüft werden, ob der jeweilige Auftragsentwickler dem Auftraggeber umfassende Rechte eingeräumt hat. Manchmal sind Klauseln zu finden, die die Veröffentlichung des Entwicklungsergebnisses als OSS verbieten. Dies gilt sinngemäß auch dann, wenn mehrere Unternehmen gemeinsam vereinbaren, eine als OSS zu veröffentlichende Software zu erstellen.

Kommentiert [KT21]: Kommentar Herr Teichert: Meines Erachtens sind das nicht alle aus Sicht einer Firma relevanten Fälle.

Kommentiert [KT22]: Kommentar Kriesel: Welche weiteren Fälle sollten denn noch betrachtet werden?

[59] BGH, Urt. vom 20.11.1984 (Az. IVa ZR 104/83 – „Kartoffelpülpe“).

Sind die Rechte geklärt, hängt es vom Einzelfall und von den mit der Veröffentlichung verbundenen Zielen ab, welche OSS Lizenz geeignet ist. Die oft schwer überschaubare Anzahl der OSS-Lizenzen zeigt hier eine Stärke, denn für fast jeden Zweck gibt es bestehende OSS-Lizenzen, so dass es selten notwendig sein wird, eine individuelle OSS-Lizenz zu erstellen.

Ein verbreitetes Geschäftsmodell ist dabei die Veröffentlichung einer Software als OSS und daneben unter einer "kommerziellen" Lizenz anzubieten, wobei letztere oft mit zusätzlichen Leistungen verbunden sind. Die OSS-Lizenzen werden oft so gewählt, dass sie bestimmte Einschränkungen wie z.B. den "Copyleft Effekt" aufweisen. Will ein Anwender die Software einsetzen, aber den "Copyleft-Effekt" vermeiden, kann er die kommerzielle Lizenz erwerben. Oft zu beobachten ist auch die Lizensierung einer Software unter zwei oder mehreren OSS-Lizenzen nebeneinander, etwa, wenn Kompatibilitätsprobleme zwischen verschiedenen OSS-Lizenzen vermieden werden sollen.

Beim **Fall 2** kommt es darauf an, ob die bearbeitete OSS unter einer Copyleft-Lizenz (vgl. Ziff. 2.3.1) oder unter einer permissiven Lizenz (vgl. Ziff. 2.3.2) veröffentlicht wurde. Bei einer permissiven Lizenz ist in der Regel keine Veröffentlichung des bearbeiteten Source Codes erforderlich. So erlaubt etwa die sogenannte BSD-Lizenz die Verbreitung einer modifizierten Version der OSS, solange für diese weiterhin die BSD Lizenz gilt; eine Herausgabe des Quellcodes ist nicht erforderlich. Demgegenüber muss eine unter einer Copyleft-Lizenz stehende bearbeitete OSS auch im Source Code herausgegeben werden, zumindest auf Anfrage: "Sie müssen dafür sorgen, dass jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird" (vgl. "inoffizielle" Übersetzung der GPLv2, http://www.gnu.de/documents/gpl-2.0.de.html, dort § 2 Ziff. 2). Der Einsatz von solchen "Copyleft"-lizensierten OSS Komponenten zusammen mit proprietärer Software, deren Quellcode nicht veröffentlicht werden soll, erfordert daher einiges an Planung.

Manche OSS Lizenzen sehen vor, dass eine Überarbeitung der jeweiligen OSS, die durch einen Dritten im Auftrag und zur exklusiven Nutzung des Auftraggebers erfolgt, nicht als "Verbreitung" oder "Weitergabe" angesehen wird und somit "Copyleft" Effekte nicht ausgelöst werden. Bekanntestes Beispiel dürfte die GPLv3 sein, die vorsieht: "Sie dürfen betroffene Werke an Dritte übertragen für den einzigen Zweck, Modifikationen exklusiv für Sie durchzuführen oder Einrichtungen für Sie bereitzustellen, um diese Werke auszuführen, vorausgesetzt, Sie erfüllen alle Bedingungen dieser Lizenz für das Übertragen von Material, dessen Urheberrecht nicht bei Ihnen liegt. Diejenigen, die auf diese Weise betroffene Werke für Sie anfertigen oder ausführen, müssen dies ausschließlich in Ihrem Namen tun, unter Ihrer Anleitung und Kontrolle und unter Bedingungen, die ihnen verbieten, außerhalb ihrer Beziehung zu Ihnen weitere Kopien Ihres urheberrechtlich geschützten Materials anzufertigen." (vgl. "inoffizielle" Übersetzung der GPLv3, http://www.gnu.de/documents/gpl.de.html, dort Ziff. 2, 2. Absatz)

Der **Fall 3** unterscheidet sich insofern vom Fall 2, als dass der Bearbeiter der OSS die bearbeitete OSS nicht (nur) selbst veröffentlichen möchte, sondern daran interessiert ist, die Änderung möglichst weit und möglichst schnell zu verbreiten, und dies mit möglichst geringem Aufwand. Um dies zu erreichen, kann eine Überlassung der Bearbeitung an den ursprünglichen Rechteinhaber der OSS mit einer umfassenden Rechtseinräumung an diesen verbunden werden. Viele Entwickler von OSS haben sich zu diesem Zweck eigene Regelungen geschaffen, um solche „Contributions“ genannten Eingaben auf eine formalrechtliche Grundlage zu heben. Diese Vereinbarungen dienen ähnlich den OSS-Lizenzen selbst dem Zweck, den Entwicklern umfassende Rechte an der angebotenen Bearbeitung zu verschaffen. Zu Einzelheiten siehe Ziffer 6.12.

6.10.2 Contribution Agreement

Für die Entwicklung freier Software ist die Mitarbeit vieler, oftmals freier Programmierer typisch. Manche der resultierenden Werke werden unabhängig voneinander geschrieben und sind unabhängig voneinander verwertbar, andere wiederum werden im Rahmen eines gemeinsamen Projekts erstellt

und verfolgen dabei die Entwicklung einer Gesamtidee. Erstere können gemäß §9 UrhG miteinander verbunden werden (verbundenes Werk), und jeder der Programmierer bleibt Urheber des von ihm geschaffenen Werkes. Letztere, hingegen, gelten gemäß §8 UrhG als Miturheber, die eine Gesamthandschaft bilden und stets gemeinsam über die Verwertung des Werkes entscheiden müssen. Die wohl wichtigste Entscheidung, die die Gesamthandschaft treffen muss, ist unter welcher Open-Source-Lizenz, z.B. GPL, Apache- oder BSD, der Code freigegeben werden soll.

Das erscheint erst einmal nicht problematisch, da ein Projekt zu dem Code beigesteuert wird oftmals schon unter einer bestimmten Open-Source-Lizenz steht.

Problematisch wird es erst, wenn ein oder vielleicht auch mehrere Programmierer glauben, dass ein Lizenzwechsel oder eine duale oder sogar multiple Lizenzierung für ein Projekt von Nutzen sein könnte.

Je mehr Programmierer beteiligt sind und je größer das Zeitfenster in dem die Entwicklungen stattfinden, desto schwieriger ist es, ein Projekt effizient zu steuern, da die Kommunikation mit den einzelnen Autoren bisweilen unmöglich wird.

Um dieses Dilemma zu umgehen, greifen viele Projekte und Organisationen auf das Modell der treuhänderischen Rechtswahrnehmung zurück, in dem sie sich vor Übernahme eines Beitrags in den „offiziellen“ Release ein „Contributor Agreement“ zugunsten des Projekts unterzeichnen lassen. Je nach Ausgestaltung gibt der Urheber damit jegliche Rechte an seinem Werk auf (Copyright Assignment Agreement - CAA) oder räumt „nur“ (nicht-) exklusive Nutzungsrechte (Copyright License Agreement - CLA) ein.

6.10.2.1 Copyright Assignment Agreement (CAA)

Das CAA stammt aus dem amerikanischen Raum. Wichtig ist, dass das dortige Konzept von „Copyright“ nicht eins zu eins mit dem deutschen Urheberrecht vergleichbar ist. So ist es dort z.B. ohne weiteres möglich sein „Copyright“ vollständig abzutreten (Title 17 chapter 2 U.S. Code §201 (d) – US Copyright Act). Ein solcher „Transfer of Ownership“ bedeutet die absolute Aufgabe jeglicher Rechte an dem jeweiligen Werk.

Die Free Software Foundation, von Richard Stallmann ins Leben gerufen, war eine der ersten Organisationen, die ihre Entwickler dazu aufgefordert hat CAAs zu unterschreiben und ihre Rechte abzutreten.

Grund hierfür war eine effektivere Projektsteuerung. Zum einen können die Projekte nicht nur an die neuesten technischen Bedingungen angepasst werden, ohne darauf zu bauen, dass jeder Urheber erlauben muss, dass sein Werk in "jede(r) spätere(n) Lizenzversion" veröffentlicht werden darf. Zudem löst es ein Problem bei der Rechtewahrnehmung.

Nur der Inhaber ausschließlicher Rechte bzw. der Rechteinhaber ist befugt ein verletztes Recht vor Gericht durchzusetzen (§501 U.S. Copyright Act)

Denn jeder einzelne Programmierer hat zwar jedem weiteren Nutzer eine einfache Nutzungslizenz eingeräumt, bleibt jedoch Urheber und behält somit das Recht (und die Pflicht) die Verletzung an seinem Code vor Gericht durchzusetzen.

Praktisch hieße das, dass mitunter 1000 von Programmierern sich zusammentun müssten um gegen Verstöße gegen die GPL zu klagen. Das ist wenig realistisch. Durch die Bündelung der Rechte in der FSF ist diese aktiv legitimiert und kann verletzte Rechte ahnden.

Das Gleiche gilt für Unternehmen. Auch wenn die Open-Source-Software-Bewegung ihren Ursprung in der Welt der Freiwilligen genommen hat, so gibt es heutzutage durchaus Unternehmen, die sich für die Entwicklung von Open-Source-Software öffnen und dabei durchaus auch auf strenge Copyleft-Lizenzen (d.h. die GPL oder die AGPL) zurückgreifen, um einen Wettbewerbsvorteil zu sichern.

Probleme bei der Urheberrechteabtretung an Unternehmen:

Vielfach stößt eine reine Rechteabtretung an ein Unternehmen aber auf Unmut.

Die Gründe hierfür sind mannigfaltig:

1. In manchen Ländern, z.B. in Deutschland, ist eine reine Rechteabtretung unmöglich, da der Urheber seine Rechte nicht abtreten kann.

2. Das andere Unternehmen dessen Entwickler den Code gestellt hat, will den Code nicht abtreten, da es eventuell anderweitig kommerziell genutzt werden könnte.

3. Das Unternehmen könnte den Code unter einer anderen weniger strengen Open-Source-Lizenz veröffentlichen oder sogar in proprietären Code einbinden.

4. Das Unternehmen könnte das Freie und Open-Source-Projekt ganz in ein proprietäres Projekt umwandeln und die externen Entwickler den Zugang zu ihrem Werk verlieren.

6.10.2.2 Copyright License Agreement (CLA)

Wenn eine reine Rechtabtretung in Form eines CAA rechtlich nicht möglich, oder nicht durchsetzbar ist, greifen viele Unternehmen, aber auch viele Foundations, zu einem Copyright License Agreement (CLA)

Durch ein CLA räumt der Urheber dem Projekt, der Organisation oder dem Unternehmen einfache oder ausschließliche Nutzungsrechte ein, die je nach Ausgestaltung des jeweiligen CLAs über die Freie oder Open-Source-Lizenz hinausgehen.

Mit anderen Worten: während ein CAA dem Rechteinhaber automatisch alle Rechte des Urhebers einräumt, d.h. das Projekt zu re-lizensieren, Unterlizenzen zu gewähren oder eine mögliche Verletzung vor Gericht zu verfolgen, tun CLAs das nur, wenn sie explizit solche Rechte enthalten.

Besonders bekannt ist das Apache CLA.

6.10.3 Arbeitsrecht

6.10.3.1 Urheberrechte und Vergütungsansprüche von Arbeitnehmern

Für die Rechtsinhaberschaft an Open-Source-Open-Source-Software gelten in einem Arbeitsverhältnis zunächst die allgemeinen gesetzlichen Regeln (§§ 69a ff. UrhG).

Bei der Entwicklung von Open-Source-Open-Source-Software durch einen Arbeitnehmer in Wahrnehmung seiner arbeitsvertraglichen Aufgaben besteht eine zentrale Besonderheit. Nach den gesetzlichen Regeln (§ 69b Abs. 1 UrhG) bleibt der Arbeitnehmer zwar Miturheber oder Urheber der Software. Dem Arbeitgeber stehen jedoch alle vermögensrechtlichen Befugnisse zu und damit auch die ausschließlichen Nutzungsrechte, sofern nichts anderes vereinbart ist (vgl. zu den Formen der Urheberschaft und der Rechtsinhaberschaft bereits oben 6.5.2). Nach mittlerweile überwiegender Meinung gilt die gesetzliche Rechtszuordnung nicht nur allgemein für Open-Source-Open-Source-Software sondern insbesondere auch für Open-Source-Open-Source-Software unter dem GPL-Lizenzmodell. Die vom Arbeitgeber an einen Entwickler im Rahmen des Arbeitsvertrages gezahlte Vergütung stellt insoweit keine vom GPL-Lizenzmodell verbotene Lizenzgebühr für eine Rechtseinräumung durch den Arbeitnehmer als Urheber dar.

Neben den besonderen Regelungen des § 69b Abs. 1 UrhG gelten die allgemeinen Regelungen des Urheberrechts. Die Entwicklung einer Open-Source-Open-Source-Software durch einen Arbeitnehmer kann rechtlich auch als Miturheberschaft, Werkverbindung oder als Bearbeitung einzuordnen sein, sodass dem Arbeitnehmer in diesen Fällen auch die Vermögensrechte aus dem Urheberrecht zustehen. Wenn eine Bearbeitung im urheberrechtlichen Sinne vorliegt, ist also der Arbeitgeber Inhaber der wirtschaftlichen Befugnisse aus dem Bearbeiter-Urheberrecht (§ 3 UrhG). Und soweit eine Miturhebergemeinschaft vorliegt (§ 8 UrhG) stehen dem Arbeitgeber die wirtschaftlichen Rechte aus

dem Miturheberanteil des bei ihm beschäftigten Arbeitnehmers zu. Als Folge von § 69b Abs. 1 UrhG steht es dem Arbeitgeber, nicht dem Arbeitnehmer, zu, über die wirtschaftlichen Verwertungsrechte an der Software zu verfügen, etwa durch Abschluss eines Contribution Agreements (vgl. Ziffer 6.12.1)..

§ 69b UrhG selbst verpflichtet den Arbeitgeber im Übrigen nicht, über die arbeitsvertragliche Vergütung hinaus eine Vergütung für diese gesetzliche Rechtszuordnung zu bezahlen.

Die gesetzlichen Besonderheiten gelten über Arbeitsverhältnisse hinaus entsprechend auch für Beamte und Beschäftigte des öffentlichen Dienstes (vgl. § 69b Abs. 2 UrhG).

6.10.3.2 Vertragliche Gestaltungsempfehlungen

Die gesetzlichen Regeln bieten dem Arbeitgeber nur scheinbar eine umfassende Sicherheit für die Rechtsinhaberschaft. Im Detail sind zahlreiche praktisch äußerst relevante Fragen zum Umfang der eingeräumten Nutzungsrechte noch nicht hinreichend verlässlich geklärt. So ist durchaus umstritten, ob eine nur in Deutschland geltende Rechtsnorm (§ 69b Abs. 1 UrhG) dem Arbeitgeber überhaupt Nutzungsrechte außerhalb Deutschlands einräumen kann und damit etwa eine weltweite Nutzung ermöglicht. Vor diesem Hintergrund ist aus Arbeitgebersicht **dringend zu empfehlen**, die Rechtsinhaberschaft und die Einräumung von Nutzungsrechten umfassend und ausdrücklich im Arbeitsvertrag zu vereinbaren. Weiterer Vorteil einer solchen Vereinbarung ist auch, dass dort möglicherweise über die Arbeitsvergütung hinausgehende gesetzliche Vergütungsansprüche aus Urheberrecht (§§ 32, 32a UrhG) für die Verwendung als Open-Source-Software weitgehend rechtssicher geregelt und wohl auch ausgeschlossen werden können.

Beim Arbeitgeber können Tarifverträge oder Betriebsvereinbarungen anwendbar sein und diese Regelungen zur Rechtszuordnung und Vergütung enthalten. Dann ist bei arbeitsvertraglichen Regelungen zur Rechtszuordnung unbedingt darauf zu achten, Widersprüche zu den Tarifverträgen oder Betriebsvereinbarungen zu vermeiden. Liegen Widersprüche vor, gilt die für den Arbeitnehmer jeweils günstigere Regelung (sog. Günstigkeitsprinzip).

Die Empfehlung einer ausdrücklichen Vereinbarung im Anstellungsvertrag gilt im Übrigen ganz besonders auch gegenüber Personen außerhalb von Arbeits- oder öffentlich-rechtlichen Dienstverhältnissen. Hier ist insbesondere an Personen mit Organstellung (z. B. Geschäftsführer oder Vorstandsmitglieder) oder (echte) freie Mitarbeiter zu denken. Die Regelung des § 69b UrhG findet für diese Personen von vornherein **keine Anwendung**. Ohne vertragliche Regelung verbleiben die Urheberstellung und die Nutzungsrechte deshalb jedenfalls zunächst bei diesen Personen.

6.10.3.3 Arbeitnehmererfindungsrecht

Open-Source-Software kann parallel zum Urheberrechtsschutz als Implementierung eines Verfahrens technischer Natur auch dem Patentschutz unterliegen (dazu bereits oben 6.8.1). In Arbeitsverhältnissen richten sich die patentrechtliche Rechtszuordnung und die Vergütung nach dem Arbeitnehmererfindungsgesetz (ArbNErfG). Ist die patentfähige Open-Source-Software während des Arbeitsverhältnisses und aus der dem Arbeitnehmer im Betrieb/der öffentlichen Verwaltung obliegenden Tätigkeit entstanden oder beruht sie auf Erfahrungen oder Arbeiten des Betriebes/der öffentlichen Verwaltung, stellt sie eine sog. Diensterfindung dar (vgl. § 4 Abs.2 ArbNErfG). Der Arbeitnehmer muss die Diensterfindung unverzüglich dem Arbeitgeber melden. Der Arbeitgeber kann die Diensterfindung dann durch Erklärung in Anspruch nehmen. Diese Inanspruchnahme gilt als erklärt, wenn der Arbeitgeber die Diensterfindung nicht innerhalb von vier Monaten nach einer ordnungsgemäßen Meldung freigibt (vgl. § 6 ArbNErfG). Mit anderen Worten, führt eine Untätigkeit des Arbeitgebers zur Rechtszuordnung an ihn. Die Rechtszuordnung bedeutet die Zuordnung aller vermögenswerten Rechte, lediglich das Recht auf Erfinderbenennung verbleibt dem Arbeitnehmer. Der Arbeitnehmer hat über die arbeitsvertragliche Vergütung hinaus Anspruch auf eine angemessene Vergütung, die sich in der Praxis häufig an den Lizenzgebühren für eine freie Erfindung – dem Gegenstück zur Diensterfindung – orientiert.

Kommentiert [KT23]: Kommentar Kriesel: Aufgrund des Kommentars von Herrn Teichert gestrichen.

Die Regelungen des Arbeitnehmererfindungsgesetzes sind gegenüber Arbeitnehmern nicht im Voraus abdingbar. Eine Regelung im Arbeitsvertrag bringt dem Arbeitgeber deshalb keinen Vorteil. Dringend anzuraten sind entsprechende vertragliche Regelungen aber wiederum gegenüber Personen mit Organstellung und freien Mitarbeitern. Für diese gilt das Arbeitnehmererfindungsgesetz nicht. Sie erwerben ohne vertragliche Regelung vielmehr die Rechte selbst.

6.11 Open-Source-Software und öffentliche Vergabe

6.11.1 Ausgangssituation

Open-Source-Software hat nicht nur im privatwirtschaftlichen Sektor eine bedeutende Rolle erlangt. Auch im Bereich der öffentlichen Hand findet Open-Source-Software mehr und mehr Verwendung. Dies spiegelt sich nicht nur in Empfehlungen von Behörden und IT-Zentren des Bundes zur Nutzung von Open-Source-Software wider, sondern auch in der Tatsache, dass viele IT-Vorhaben im Öffentlichen Bereich bereits unter Verwendung von Open-Source-Software umgesetzt werden, ja sogar im Kern auf diesen basieren. Als aktuelles Beispiel kann die Entwicklung der Zahlungsverkehrsplattform ePayment Bund/Länder (ePayBL) genannt werden. Diese stellt Bundes- und Landesbehörden als Betreiber von Internetanwendungen wie E-Shops oder Vorgangsbearbeitungssystemen eine Anbindung an das Haushaltssystem des Bundes zur Verfügung. Hierbei wird ein barrierefreier (Muster-)Webshop auf Basis von Open-Source-Software entwickelt, der eine einheitliche Lösung für eine behördenindividuelle Web-Präsenz zur Darstellung und zum Erwerb von Behördenleistungen mit integriertem ePayment bereitstellen soll.[60]

Auch auf EU-Ebene sind bereits seit 2008 Maßnahmen ergriffen worden, um die Einbeziehung von Open-Source-Software im Rahmen der öffentlichen Auftragsvergabe zu fördern.[61] Das Open-Source-Observatory and Repository (OSOR) hat in diesem Zusammenhang einen einschlägigen Leitfaden erstellt, der unter anderem rechtliche Rahmenbedingungen darstellt und fertige Textbausteine für die Ausschreibung zur Verfügung stellt.[62]

Der nachweislich weit verbreiteten Akzeptanz und Nutzung von Open-Source-Software in der öffentlichen Verwaltung stehen jedoch rechtliche Herausforderungen gegenüber. Im Bereich der öffentlichen Verwaltung geht es dabei nicht nur um Urheber- und Lizenzrechte sowie Haftungs- und Gewährleistungsrechte. Soweit sich die öffentliche Hand zur Beschaffung von Open-Source-Software entschließt, kommt eine weitere Rechtsebene hinzu, nämlich die des Vergaberechts.

Die nachfolgenden Ausführungen sollen einen Überblick über die vergaberechtlichen Besonderheiten bei der Beschaffung von Open-Source-Software geben. Hierbei soll zunächst auf die Frage eingegangen werden, inwieweit bei der Beschaffung von Open-Source-Software die vergaberechtlichen Vorschriften überhaupt Anwendung finden (siehe Ziffer 6.9.2). Sodann ist zu klären, inwieweit öffentliche Auftraggeber konkrete Vorgaben bezüglich der Einbindung oder des Ausschlusses von Open-Source-Software bzw. proprietärer Software machen dürfen (dazu Ziffer 6.9.3). Daran anschließend soll dargelegt werden, welcher Konflikt bei einer wettbewerbsoffenen Ausschreibung zwischen der Verwendung von Open-Source-Software und den vergaberechtlichen Bestimmungen besteht und wie dieser gelöst werden kann (vgl. Ziffer 6.9.4). Schließlich werden noch einmal die wesentlichen Schritte der Beschaffung von Open-Source-Software aufgezeigt (vgl. Ziffer 6.9.5).

---

[60] So beschrieben in „Digitale Verwaltung 2020 – Regierungsprogramm 18. Legislaturperiode“ (September 2014), S. 24

[61] Die aktuelle Open-Source-Strategie der EU-Kommission 2014 – 2017 ist hier veröffentlicht: http://ec.europa.eu/dgs/informatics/oss_tech/strategy/strategy_en.htm

[62] Der Leitfaden steht – soweit ersichtlich – nur in englischer Sprache zur Verfügung: Guideline on public procurement of Open Source Software (März 2010)

6.11.2 Vergaberechtliche Bindung bei der Beschaffung von Open-Source-Software

Bei jeder Leistungserbringung für öffentliche Auftraggeber besteht der „Anfangsverdacht“, es handelt sich um einen ausschreibungspflichtigen Vorgang. Vor diesem Hintergrund ist zu untersuchen, in welchem Umfang die Beschaffung von OSS ebenfalls den vergaberechtlichen Regelungen unterfällt (dazu 6.9.2.2). Zuvor sollen jedoch kurz die unterschiedlichen Beschaffungskonstellationen im Zusammenhang mit Open-Source-Software aufgezeigt werden (dazu 6.9.2.1).

6.11.2.1. Beschaffungskonstellationen

In Abhängigkeit vom konkreten Beschaffungsbedarf des öffentlichen Auftraggebers kann Open-Source-Software in verschiedenen Konstellationen Gegenstand einer Ausschreibung sein. Zum einen kann sich die Ausschreibung auf die „reine“ Überlassung von Open-Source-Software beschränken. Allerdings ist eine isolierte Beschaffung / Überlassung von Open-Source-Software in der Praxis eher die Ausnahme. In der Mehrzahl der Fälle werden neben der Überlassung der Open-Source-Software weitergehende Dienstleistungen, wie etwa nutzerspezifische Anpassungen oder Wartung und Instandhaltung der Open-Source-Software, ausgeschrieben.[63]

Von der Software-Überlassung – mit oder ohne zusätzliche Dienstleistungen – sind diejenigen Ausschreibungen zu unterscheiden, die eine nachträgliche Weiterentwicklung bereits überlassener Open-Source-Software zum Gegenstand haben. In diesen Konstellationen geht es zumeist um die Anpassung von Open-Source-Software an neue gesetzliche Vorschriften oder technische Weiterentwicklungen. Kennzeichen dieser Ausschreibungen ist mithin, dass die Beschaffung der Open-Source-Software und die nachträglichen Anpassungsleistungen zeitlich auseinanderfallen, mithin nicht in unmittelbarem Zusammenhang stehen.[64]

Zudem kann Open-Source-Software als Fremdkomponente eines (anderen) Soft- oder Hardware-Produktes beschafft werden. Hierbei ist zudem zu differenzieren, ob die Open-Source-Software integraler Bestandteil eines anderen (proprietären) Soft- oder Hardware-Produktes ist, oder ob die Open-Source-Software „lediglich“ für den Betrieb der anderen (proprietären) Soft- oder Hardware erforderlich ist. Im letztgenannten Fall kann die Open-Source-Software auch unabhängig von der (Grund-)Software laufen und ggf. separat installiert werden.[65]

6.11.2.2 Beschaffung von Open-Source-Software und Vergaberecht

Inwieweit die Beschaffung von Open-Source-Software in einer der vorgenannten Konstellationen dem Vergaberecht unterfällt, bestimmt sich einerseits nach Regelungen des EU-Vergaberechts und andererseits nach den nationalen Vergaberechtsregelungen.

Das EU-Vergaberecht findet Anwendung, wenn erstens der geschätzte Auftragswert den relevanten Schwellenwert überschreitet.[66] Zum zweiten muss der persönliche Anwendungsbereich des EU-Vergaberechts eröffnet sein; d.h., es muss sich um die Beschaffung eines öffentlichen Auftraggebers im Sinne des § 98 GWB handeln. Dabei werden nicht nur die „klassischen“ öffentliche Auftraggeber Bund, Länder und Kommunen vom Vergaberecht erfasst (vgl. § 98 Nr. 1 und Nr. 3 GWB). Im Bereich des EU-Vergaberechts ist vielmehr von einem funktionalen Auftraggeberbegriff auszugehen.[67] Sofern die Voraussetzung des § 98 Nr. 2 GWB vorliegen, müssen juristische Personen des Privatrechts

---

[63] vgl. z.B. Ausschreibung der europäischen Zentralbank über die Lieferung und Pflege von Betriebssystemsoftware für Server

[64] vgl. Ausschreibung „Aktualisierung und Erweiterung des auf Open-Source-Software erstellten Programms „dataDIVER“ als Bestandteil der Geodateninfrastruktur des Bundesamtes für Seeschifffahrt und Hydrographie

[65] Gerlach, CR 2012, 691 (692).

[66] Für Liefer- und Dienstleistungsaufträge beträgt der Schwellenwert derzeit (1. Halbjahr 2015) 207.000 EUR netto, es sei denn, es handelt sich um Ausschreibungen von obersten und oberen Bundesbehörden (dann 134.000 EUR netto) oder Sektorenauftraggebern bzw. Ausschreibungen im Verteidigungs- und Sicherheitsbereich (dann 414.000 EUR netto).

[67] EuGH, Urt. v. 15.05.2003 – C-214/00 – Vertragsverletzung Spaniens wegen einer Ausnahme für in privater Rechtsform organisierte Auftraggeber; OLG Düsseldorf v. 19.06.2013 - VII-Verg 55/12 – BWI Services GmbH

ebenfalls das Vergaberecht beachten. Und zum dritten muss es sich um einen „öffentlichen Auftrag“ im Sinne des § 99 GWB handeln (sog. sachliche Anwendungsbereich). Danach sind öffentliche Aufträge entgeltliche Verträge von öffentlichen Auftraggebern mit Unternehmen über die Beschaffung von Leistungen, die Liefer-, Bau- oder Dienstleistungen zum Gegenstand haben. Der Begriff des „entgeltlichen Vertrags“ ist weit auszulegen. Die Gegenleistung muss nicht notwendig in Geld bestehen, sondern erfasst jede Art von Vergütung, die einen Geldwert hat.[68]

Unterhalb der Schwellenwerte ergibt sich die vergaberechtliche Bindung „klassischer“ öffentlicher Auftraggeber aus dem Haushaltrecht. Grundsatznorm ist dabei § 30 HGrG.[69] Da Bund und Länder gemäß § 1 Abs. 1 HGrG verpflichtet sind, die Regelung des § 30 HGrG bei der Gesetzgebung einzuhalten, finden sich wort- oder inhaltsgleiche Regelungen auf Bundesebene in § 55 Abs. 1 BHO, auf Landesebene zum Beispiel in § 55 Abs. 1 LHO NRW, § 55 Abs. 1 SäHO oder für Kommunen in § 25 Abs. 1 GemHVO NRW. Soweit die vergaberechtliche Bindung aus dem Haushaltrecht folgt, bedarf es zudem einer finanzwirksamen Leistung. Unentgeltliche Beschaffungen klassischer öffentlicher Auftraggeber unterfallen daher auch unterhalb der EU-Schwellenwerte nicht dem Vergaberecht. Auftraggeber, die oberhalb der Schwellenwerte nach dem funktionalen Auftraggeberbegriff an das Vergaberecht gebunden wären, finden im Haushaltsrecht regelmäßig keine Berücksichtigung. Es ist daher zu prüfen, ob die Bindung derartiger Auftraggeber an das Vergaberecht aus spezialgesetzlichen Regelungen, Satzungen, Gründungsstatuten oder Fördermittelbescheiden etc. folgt.

Wendet man die vorstehenden Maßstäbe auf die Beschaffung von Open-Source-Software an, ist zwischen der „reinen“ Überlassung von Open-Source-Software auf der einen und den weitergehenden Beschaffungskonstellationen auf der anderen Seite zu unterscheiden. Umfasst die Beschaffungsmaßnahme allein die Open-Source-Software – sei es, dass die Open-Source-Software von einem Dritten kostenlos bereitgestellt wird oder, dass sich der öffentliche Auftraggeber die Open-Source-Software aus dem Internet kostenlos herunterlädt -, findet das Vergaberecht keine Anwendung. Es fehlt bei einer „reinen“ Open-Source-Software-Beschaffung an der Entgeltlichkeit im Sinne des § 99 GWB, bzw. es liegt keine finanzwirksame Leistung im Sinne des Haushaltrechts vor.

Sollen neben der Überlassung der Open-Source-Software weitergehende Dienstleistungen, wie etwa nutzerspezifische Anpassung oder Wartung und Instandhaltung der Open-Source-Software, beschafft werden, sind die weitergehenden entgeltlichen Dienstleistungen stets ausschreibungspflichtig. Ob zudem die Überlassung der zugrunde liegenden Open-Source-Software ausschreibungspflichtig ist, hängt davon ab, ob die kostenlose Softwareüberlassung und die weitergehenden entgeltlichen Dienstleistungen als separate Vorgänge oder aber als einheitlicher Vorgang betrachtet werden. Allein im Falle einer Gesamtbetrachtung stellt auch die Beschaffung der Open-Source-Software einen ausschreibungspflichtigen Vorgang dar. Unabhängig von der Tatsache, dass der Einsatz von Open-Source-Software in beiden Konstellationen begründet werden muss (siehe dazu Ziffer 6.9.3), sollten zudem die Vor- und Nachteile eines getrennten Vorgehens gegeneinander abgewogen werden. Bei der erforderlichen Abwägung ist zu berücksichtigen, dass den jeweiligen Vorgängen unterschiedliche Rechtsverhältnisse mit unterschiedlichen Haftungsregelungen zugrunde liegen. Ferner können Probleme bei der Gewährleistungsabgrenzung auftreten. Letztlich wird es auf die zu ermittelnden Umstände des Einzelfalls ankommen, welcher Weg eingeschlagen wird.

Sofern die Beschaffungsmaßnahme die nachträgliche Weiterentwicklung bereits überlassener Open-Source-Software zum Gegenstand hat, finden die vergaberechtlichen Vorschriften in jedem Fall Anwendung. Grund hierfür ist, dass der öffentliche Auftraggeber in diesen Konstellationen allein entgeltliche Dienstleistungen bezüglich einer Open-Source-Software ausschreibt, aber nicht die Softwareüberlassung an sich. Die Frage nach einer Gesamtbetrachtung stellt sich mithin nicht.

Ferner unterliegt die Beschaffung von Open-Source-Software in Form von Fremdkomponenten grundsätzlich dem Vergaberecht. Auch in diesen Beschaffungskonstellationen ist nicht die Open-

[68] Zeiss in: Heiermann / Zeiss, jurisPK-Vergaberecht, 4. Aufl. 2013, § 99 GWB, Rn. 103

[69] Wortlaut dieser Vorschrift: „Dem Abschluß von Verträgen über Lieferungen und Leistungen muß eine öffentliche Ausschreibung vorausgehen, sofern nicht die Natur des Geschäfts oder besondere Umstände eine Ausnahme rechtfertigen.“

Source-Software an sich Beschaffungsgegenstand, sondern die (proprietären) Soft- und Hardwareprodukte, in die Open-Source-Software als ein „Baustein“ integriert ist. Etwas anderes könnte allenfalls bei sog. „Stand-Alone-Komponenten“ gelten. Hier sind Fälle denkbar, bei denen die Überlassung der kostenlosen Open-Source-Software von den weiteren Liefer- oder Dienstleitungen getrennt betrachtet werden kann mit der Folge, dass die Softwareüberlassung ggf. nicht ausschreibungspflichtig ist. Diese Ausnahmefälle gleichen mithin den Konstellationen, bei denen neben der Überlassung der Open-Source-Software weitergehende Dienstleistungen, wie etwa nutzerspezifische Anpassung oder Wartung und Instandhaltung der Open-Source-Software, beschafft werden.

6.11.3 Bestimmungsfreiheit des öffentlichen Auftraggebers

Im Zusammenhang mit der Beschaffung von Open-Source-Software stellt sich ferner die Frage, ob öffentliche Auftraggeber in den Vergabeunterlagen, insbesondere in der Leistungsbeschreibung, konkrete Vorgaben hinsichtlich der Einbindung oder aber des Ausschlusses von Open-Source-Software machen dürfen. Aus vergaberechtlicher Sicht ist dies insoweit von Bedeutung, als die Entscheidung für oder gegen Open-Source-Software stets mit einem Wettbewerbsnachteil entweder für proprietäre Software oder für Open-Source-Software verbunden ist. Im Weiteren soll daher geklärt werden, inwiefern sich öffentliche Auftraggeber bereits im Vorfeld der Ausschreibung auf einen technischen Standard festlegen dürfen.

Ausweislich § 97 Abs. 1 GWB sind öffentliche Auftraggeber verpflichtet, Waren sowie Bau- und Dienstleistungen im Wettbewerb zu beschaffen. Öffentliche Auftraggeber sind danach gehalten, Ausschreibungen wettbewerbsoffen und diskriminierungsfrei zu gestalten (§ 2 Abs. 1 VOL/A). Dementsprechend regeln §§ 7 Abs. 3 VOL/A, 8 EG Abs. 7 VOL/A das Gebot der produktneutralen Ausschreibung.[70] Aus dem vergaberechtlichen Gebot der produktneutralen Ausschreibung kann jedoch nicht gefolgert werden, dass der öffentliche Auftraggeber in seiner Entscheidung, welchen Auftragsgegenstand er für erforderlich hält, gebunden ist. Das Vergaberecht regelt nämlich nicht, was der öffentliche Auftraggeber beschafft, sondern nur die Art und Weise der Beschaffung.[71] Der öffentliche Auftraggeber kann daher grundsätzlich frei darüber befinden, was für ein Produkt oder Verfahren oder dergleichen er beschafft. Die Wahl unterliegt der Bestimmungsfreiheit des Auftraggebers, deren Ausübung dem Vergabeverfahren vorgelagert ist.[72]

Die Bestimmungsfreiheit des öffentlichen Auftraggebers hinsichtlich des Beschaffungsgegenstandes ist jedoch nicht grenzenlos. Sie unterliegt bestimmten, durch das Vergaberecht gezogenen Grenzen. Diese Grenzen sind eingehalten, sofern

- die Bestimmung durch den Auftragsgegenstand sachlich gerechtfertigt ist,
- vom Auftraggeber dafür nachvollziehbare, objektive, auftragsbezogene und tatsächlich Gründe angegeben worden sind und
- die Bestimmung andere Wirtschaftsteilnehmer nicht diskriminiert.[73]

Die Rechtsprechung zur Bestimmungsfreiheit öffentlicher Auftraggeber lässt sich ohne weiteres auf die eingangs gestellte Frage übertragen, ob konkrete Vorgaben hinsichtlich der Einbindung oder aber des Ausschlusses von Open-Source-Software zulässig sind. Sach- und auftragsbezogene Gründe können sich vor allem aus wirtschaftlichen Erwägungen, Sicherheitsgedanken sowie Anpassungs- und Integrationsmöglichkeiten ergeben. Für Open-Source-Software sprechen vor allem wirtschaftliche Erwägungen. Zum einen ist sie kostenlos (oder zu geringen Kosten in Alternativen Bezugsmodellen

[70] Soweit es nicht durch den Auftragsgegenstand gerechtfertigt ist, darf in den technischen Anforderungen nicht auf eine bestimmte Produktion oder Herkunft oder besondere Verfahren oder Marken, Patente oder Typen, einen bestimmtem Ursprung oder bestimmte Produktionen verwiesen werden, wenn dadurch bestimmte Unternehmen oder bestimmte Produkte begünstigt oder ausgeschlossen werden.
[71] OLG Düsseldorf, Beschluss v. 01.08.2012 – VII Verg. 10/12
[72] OLG Düsseldorf, Beschluss v. 22.05.2013 – VII Verg. 16/12
[73] so zuletzt OLG Jena, Beschluss v. 25.06.2014 – 2 Verg. 1/14

zu beschaffen). Zum anderen besteht bei Open-Source-Software in der Regel keine Notwendigkeit zur kostenpflichtigen Pflegevereinbarung mit einem Hersteller. Ferner ist aufgrund der offenen Standards die spätere Ablösung oftmals problemloser und kostengünstiger als bei proprietären Lösungen.[74] Allerdings ist im Einzelfall zu untersuchen, ob die Kostenvorteile ggf. durch erhöhte Kosten für die Migration und Mitarbeiterschulungen wieder ausgeglichen werden.

Selbst eine Beschaffung im sicherheitsrelevanten Bereich muss nicht zwangsläufig gegen einen Einsatz von Open-Source-Software sprechen. Ebensowenig spricht die Notwendigkeit einer langfristigen Pflege gegen den Einsatz von Open-Source-Software, Denn Open-Source-Software kann gerade auch durch andere als die ursprünglichen Autoren gepflegt werden.

Andererseits kann eine Beschränkung auf Open-Source-Software sachlich gerechtfertigt sein, wenn die Weiterentwicklung der Software den Mitarbeitern des öffentlichen Auftraggebers übertragen werden soll. Aber auch insoweit gilt, dass die konkreten Umstände des Einzelfalls entscheidend sind. Eine pauschale Entscheidung ist weder für noch gegen die Einbindung von Open-Source-Software zulässig.

Ebenso können sich sach- und auftragsbezogene Gründe aus den technischen Gegebenheiten ergeben. Hiervon sind vor allem jene Fälle softwarebasierter Beschaffungen umfasst, in denen Softwareprodukte in ein bestehendes System integriert werden müssen. In diesen Fällen ist zu fragen, inwieweit die Einführung alternativer softwarebasierter Produkte ggf. zu unverhältnismäßig höheren Kosten, Risiken oder Schwierigkeiten bei der Nutzung oder Pflege führen würden.

Soweit sach- und auftragsbezogene Gründe vorliegen, ist es dem öffentlichen Auftraggeber mithin unbenommen, in den Vergabeunterlagen, insbesondere in der Leistungsbeschreibung, konkrete Vorgaben hinsichtlich der Einbindung oder aber des Ausschlusses von Open-Source-Software bzw. proprietärer Software zu machen. Entsprechende Vorgaben stellen keinen Verstoß gegen das Gebot der produktneutralen Ausschreibung im Sinne der §§ 7 Abs. 3 VOL/A, 8 EG Abs. 7 VOL/A dar, sofern es sich um sachlich gerechtfertigte Ausnahmefälle handelt. Im Interesse eines reibungslosen Verfahrensablaufes empfiehlt es sich, den Bietern die vorab ermittelten Gründe mitzuteilen. Dies spart nicht nur Zeit, sondern schließt unnötige Vergaberügen aus. Darüber hinaus sollte die Entscheidung umfassend in der Vergabeakte dokumentiert werden.

6.11.4 Wettbewerbsoffene Ausgestaltung des Vergabeverfahrens

Sofern keine sach- und auftragsbezogenen Gründe vorliegen, sind öffentliche Auftraggeber gehalten, die Leistungen wettbewerbsoffen auszuschreiben. Software basierte Ausschreibungen sind mithin so auszugestalten, dass Bieter sowohl Open-Source-Software als auch proprietäre Software anbieten können und dabei keine technische Lösung von vorne herein benachteiligt wird. Andererseits kann daraus nicht geschlossen werden, dass die unterschiedlichen technischen Ansätze und damit zusammenhängenden Sach- und Rechtsfolgen nicht auch unterschiedlich berücksichtigt werden dürfen. Allerdings sind öffentliche Auftraggeber verpflichtet, die Ausschreibungsunterlage so zu gestalten, dass die eingehenden Angebote – und zwar unabhängig von deren technisch-inhaltlichen Ausgestaltung – sowohl in qualitativer als auch in preislicher Hinsicht vergleichbar sind. So können beispielsweise unterschiedliche Kostenbestandteile, wie etwa Kosten für Upgrades, Updates, Migration, Schulung und Wartungs- und Pflegeleistungen separat ausgewiesen und ggf. unterschiedlich gewichtet werden. Zudem bietet es sich an, die Offenlegung des Quellcodes als Bewertungskriterium auszuweisen und besonders zu gewichten.

Ferner ist im Rahmen einer wettbewerbsoffenen Ausschreibung darauf zu achten, dass die Vertragsbedingungen die lizenz- und nutzungsrechtlichen Besonderheiten von Open-Source-Software widerspiegeln. Dies ist nicht nur mit Blick auf den Gleichbehandlungsgrundsatz geboten. In diesem Zusammenhang sind ferner §§ 13 Abs. 4, 16 EG Abs. 4 Satz 1 VOL/A von Bedeutung, wonach Änderungen an den Vertragsunterlagen unzulässig sind. Der Begriff der Änderungen ist dabei weit zu

[74] vgl. insoweit OLG Düsseldorf, Beschluss v. 22.05.2013 – VII Verg 16/12

verstehen. Danach ist es Bietern untersagt, ihrem Angebot andere, als vom Auftraggeber vorgegebene Vertragsbedingungen beizufügen. Missachtet ein Bieter das Änderungsverbot, ist sein Angebot gemäß §§ 13 Abs. 3 lit. d), 19 EG Abs. 3 lit. d) VOL/A von der Wertung auszuschließen. Insbesondere bei software-bezogenen Beschaffungen von Bundesbehörden kann dies ein Problem darstellen, da diese gemäß Ziffer 3.1.1 der VV zu § 55 BHO verpflichtet sind, ihren Ausschreibungen die EVB-IT zugrunde zu legen.[75] Zwar enthalten die EVB-IT System, EVB-IT Systemlieferung sowie bereits eine Nutzungsrechtsmatrix, die es den öffentlichen Auftraggebern ermöglicht, unterschiedliche Lizenzmodelle und Nutzungsrechtsbeschränkungen im Rahmen einer Ausschreibung zu berücksichtigen. Allerdings werden in den jeweiligen Nutzungsrechtsmatrizen die Besonderheiten von Open-Source-Software, wie zum Beispiel Copyleft-Klauseln und Bearbeitungsrechte, bislang nicht abgebildet . Sofern eine Ausschreibung auf Basis von EVB-IT erfolgt und der Auftraggeber die rechtlichen Besonderheiten der Open-Source-Software nicht in anderer Form berücksichtigt, müssten Angebote mit Open-Source-Softwareanteilen, vor allem in Form von Fremdkomponenten, grundsätzlich von der Wertung ausgeschlossen werden.

Kommentiert [KT24]: Anmerkung DW: noch prüfen, welche der EVB-IT noch eine Nutzungsrechtsmatrix enthalten

Solange die EVB-IT den Besonderheiten der Lizenz- und Nutzungsbedingungen von Open-Source-Software nicht gerecht werden, ist es daher erforderlich, dass die jeweiligen EVB-IT um spezifische Regelungen zu Open-Source-Software ergänzt werden.[76] Die konkrete Ausgestaltung hängt dabei von den Umständen des jeweiligen Einzelfalls ab. Ebenso ist mit Blick auf die Besonderheiten der jeweiligen Ausschreibung zu prüfen, ob die unterschiedlichen vertraglichen Regelungen im Rahmen der Angebotswertung besonders berücksichtigt werden müssen. Denkbar ist zum Beispiel, die softwarebezogene Aspekte als Bewertungskriterien auszuweisen und zu gewichten.

Ferner besteht die Möglichkeit, dass in den Vertragsunterlagen Öffnungsklauseln für Open-Source-Software aufgenommen werden. Mittels derartiger Öffnungsklauseln erklärt der öffentliche Auftraggeber die Einbeziehung von Open-Source-Lizenzen für zulässig.[77] Darüber hinaus kann der öffentliche Auftraggeber Nebenangebote zulassen. Öffentliche Auftraggeber haben jedoch insoweit zu berücksichtigen, dass die Wertung von Nebenangeboten die Festlegung von Mindestanforderungen voraussetzt (§ 9 EG Abs. 5 VOL/A i.V.m. § 19 EG Abs. 3 lit. g) VOL/A). Zudem sind öffentliche Auftraggeber gehalten, die Vertrags- und Lizenzbedingungen der jeweiligen Open-Source-Software umfassend zu prüfen.

6.11.5 Bedarfsanalyse und Markterkundung bei Beschaffungen

Mittels der nationalen oder europaweiten Bekanntmachung machen öffentliche Auftraggeber ihre Beschaffungsabsicht publik. Die Bekanntmachung markiert den Beginn des Vergabeverfahrens im engeren Sinne. Der Bekanntmachung vorgelagert und im Hinblick auf den Erfolg des initiierten Vergabeverfahrens von wesentlicher Bedeutung sind jedoch die Bedarfsanalyse (siehe dazu 6.9.5.1) und die Markterkundung (siehe dazu 6.9.5.2). Zum einen entscheiden eine Bedarfsanalyse und Markterkundung darüber, ob öffentliche Auftraggeber auch das erhalten, was sie tatsächlich benötigen. Zum anderen können öffentliche Auftraggeber ohne Bedarfsanalyse und Markterkundung keinen Gebrauch von der ihnen zustehenden Bestimmungsfreiheit (siehe oben Ziffer 6.9.3) machen bzw. sind nicht in der Lage, die Ausschreibungsunterlagen wettbewerbsoffen auszugestalten (siehe oben Ziffer 6.9.4).

---

[75] Auf Landesebene sind die jeweiligen landesspezifischen Vorschriften zu beachten. In Bayern finden sich Vorgaben zur partiellen Anwendung der EVB-IT in der IT-Richtlinie für die bayerische Staatsverwaltung vom 01.01.2008 – BayITR-08. In Baden-Württemberg folgt beschränkte Anwendungsverpflichtung aus Punkt 4 VV-LHO und in NRW aus Ziffer 5 RdErl. d. Ministeriums für Inneres und Kommunales.

[76] vgl. näher Jaeger, in: Working Group Public Affairs der OSB Alliance, Handreichung zur Nutzung von EVB-IT beim Einsatz von Open Source Software

[77] vgl. näher Gerlach, CR 2012, S. 691 ff.

6.11.5.1 Bedeutung der Bedarfsanalyse

Im Rahmen der Bedarfsanalyse werden anhand der Anforderungen der Bedarfsträger Art und Qualität der zu beschaffenden Leistungen festgelegt. Im Falle softwarebasierter Beschaffungen geht es als also um die Ermittlung der notwendigen Funktionalitäten und Anwendungen der Software bzw. des softwarebasierten Gesamtsystems auf Basis des konkreten Bedarfs, der tatsächlichen und technischen Rahmenbedingungen. Zur Bedarfsanalyse zählen aber auch Wirtschaftlichkeitsaspekte, wie etwas Anschaffungs- oder Zusatz- und Folgekosten. Die Bedarfsanalyse ist mithin entscheidende Grundlage dafür, damit öffentliche Auftraggeber von der ihnen zustehenden Bestimmungsfreiheit Gebrauch machen können. Allein eine nachhaltige Bedarfsanalyse fördert diejenigen sach- und auftragsbezogene Gründe zutage, die ggf. konkrete Vorgaben hinsichtlich der Einbindung oder aber des Ausschlusses von Open-Source-Software bzw. proprietärer Software möglichen machen. Sach- und auftragsbezogene Gründe können sich dabei vor allem aus wirtschaftlichen Erwägungen Anpassungs- und Integrationsmöglichkeiten sowie Sicherheitsaspekten ergeben.

6.11.5.2 Markterkundung

Neben der Bedarfsanalyse kommt der Markterkundung eine besondere Rolle im Vorfeld eines Vergabeverfahrens zu. Unter Markterkundung versteht man dabei die systematische Ermittlung des aktuellen und zukünftigen Lieferangebots. Den vergaberechtlichen Vorschriften kann dabei kein konkretes Verfahren entnommen werden, welches von den öffentlichen Auftraggebern einzuhalten ist. Es sind lediglich die Grundsätze des Wettbewerbs, der Gleichbehandlung sowie der Transparenz einzuhalten. Was dies im Einzelfall bedeutet, wird jedoch nicht immer einheitlich gesehen. Vor allem im Zusammenhang mit der Bestimmungsfreiheit ist fraglich, inwieweit öffentliche Auftraggeber besondere Markterkundungsmaßnahmen durchführen und sich einen breiten Überblick über sämtliche in Betracht kommenden technischen Lösungen verschaffen müssen.[78]

Soweit konkrete Vorgaben hinsichtlich der Einbindung oder aber des Ausschlusses von Open-Source-Software bzw. proprietärer Software gemacht werden sollen, empfiehlt es sich jedoch bereits mit Blick auf den erforderlichen Nachweis der sach- und auftragsbezogenen Gründe in jedem Fall eine umfassende Markterkundung durchzuführen. Nur so ist gewährleistet, dass dem möglichen Einwand, die Bestimmung des Auftragsgegenstandes diskriminiere Anbieter von Open-Source-Software bzw. proprietärer Software, angemessen begegnet werden kann und die Entscheidung im Ergebnis auch einer Überprüfung durch die Vergabekammern und Oberlandesgerichte standhält.

Ebenso setzt eine wettbewerbsoffene Ausschreibung eine umfassende Markterkundung voraus. Nur auf diese Weise ist gewährleistet, dass öffentliche Auftraggeber Kenntnis von den unterschiedlichen Nutzungs- und Lizenzbedingungen von Open-Source-Software und proprietärer Software erlangen und diese entsprechend in den Ausschreibungsunterlagen berücksichtigen.

6.12 Open-Source-Software im Spiegel der Rechtsprechung

Die ersten Gerichtsverfahren in **Deutschland** zu Open-Source-Software in den Jahren 2000 – 2002 drehten sich um die widerrechtliche Nutzung von Marken[79]. So erwirkte die Firma Crayon im Jahr 2002 eine einstweilige Verfügung gegen den Linux-Distributor SuSE, da dieser mit einer ähnlich

[78] so zumindest OLG Celle v. 22.05.2008 – Verg 1/08, a.A. OLG Düsseldorf v. 01.08.2012 – Verg 10/12, OLG Jena v. 25.06.2014 – 2 Verg 1/14

[79] Ulrich Wuermeling / Thies Deike, „Open-Source-Source-Software: Eine juristische Risikoanalyse“; in: Computer und Recht 2 / 2003, S.87; siehe auch http://www.ifross.org/en/node/400

lautenden Open-Source-Software Krayon warb, obwohl diese nicht einmal auf der Distributions-CD enthalten war[80]. Der Streit wurde außergerichtlich beigelegt.

Das erste Urteil, in dem die Wirksamkeit der GPL in Deutschland anerkannt wurde, erließ im Jahr 2005 das Münchner Landgericht I im Rechtsstreit H.Welte gegen Sitecom zur Nutzung der Software netfilter / iptables.[81] Das Gericht setzte sich grundsätzlich mit der rechtlichen Gültigkeit der GPL auseinander und kam zu dem Schluss, diese wie allgemeine Geschäftsbedingungen zu werten.[82] Es stellte fest, dass eine Verletzung der GPL eine Urheberrechtsverletzung darstellt. An der vom Kläger erwirkten einstweiligen Verfügung wurde festgehalten. Die Beklagte hatte die Open-Source-Software in der Firmware ihres wireless Routers integriert, dies jedoch nicht offengelegt. Das Gericht verurteilte die Beklagte zur Bereitstellung der GPL-Lizenzbedingungen und des Quellcodes, sowie zur Übernahme der Prozesskosten.

Weitere Urteile, in denen eine nicht-deklarierte Nutzung von Software unter GPL geahndet wurde, folgten: im Jahr 2005 im Fall H.Welte gegen Fortinet UK durch das Landgericht München I, [83] im Jahr 2006 sowohl durch das Landgericht Berlin zur Nutzung in einem WLAN-Router,[84] als auch durch das Landgericht Frankfurt I im Fall H.Welte gegen D-Link.[85]

Im Jahr 2007 war der Beklagte Skype, der Plattformbetreiber, über den ein Produkt mit integrierter GPL-Software zum Verkauf angeboten wurde, und nicht der Ersteller des Produkts selbst. Das Landgericht München I urteilte, dass ein Verweis auf den GPL-Text und den Code unzureichend ist, und erließ eine einstweilige Verfügung.[86]

Im Jahr 2010 entschied das Oberlandesgericht Düsseldorf im Fall xt:Commerce, dass die GPL zwar urheberrechtliche Nutzungsrechte einräumt, aber keine Rechte an Marken.[87] Die Weiterverbreitung der unter GPL stehenden Software durch andere ist nur nach Entfernen der Marke rechtmäßig.

Im Jahr 2010 beurteilte das Landgericht Hamburg den kostenpflichtigen Vertrieb der kostenfreien, unter einer OSS-Lizenz stehenden Software Mozilla und Thunderbird als Verstoß gegen das Markenrecht und das UWG.[88]

Im Jahr 2011 stellte das Landgericht Bochum im Fall FreeAdhocUDF gegen WISO Mein Büro 2009 fest, dass auch eine nur für Testzwecke eingebaute Software unter der LGPL deklariert werden muss, und die Verpflichtungen aus der Lizenz erfüllt werden müssen.[89] Bei Verletzung können die gesetzlich vorgesehenen Schritte „Auskunftserteilung“ und „Schadensersatz“ eingeleitet werden.

---

[80] Wuermeling / Deike a.a.O. S.87; siehe auch http://www.heise.de/newsticker/meldung/Verwirrung-um-Verfuegung-gegen-Linux-Distributor-SuSE-55271.html

[81] Urteil v. 19.05.2004, Az. 21 O 6123/04

[82] Vgl. Laurent, Philippe, “Open Source / Content Licences before European Courts”, Vortrag auf dem European Opensource & Free Software Law Event 2012 (EOLE 2012), S. 6; Van den Brande / Coughlan / Jäger (ed.), The International Free and Open-Source-Software Law Book, 2nd edition, S. 204 ff.

[83] Laurent a.a.O., S.7

[84] Urteil vom 21.02.2006, Az. 16 O 134/06; vgl. auch http://www.telemedicus.info/urteile/Urheberrecht/Open-Source-/556-LG-Berlin-Az-16-O-13406-Verstoss-gegen-GPL-WLAN-Router.html

[85] Urteil vom 06.09.2006, Az. 2-6 O 224; Philippe Laurent a.a.O., S. 8; Van den Brande / Coughlan / Jäger a.a.O. S. 206

[86] Beschluss vom 24.05.2007, Az. 7 O 5245/07; vgl. auch http://www.telemedicus.info/urteile/Urheberrecht/Open-SourceOpen-Source-/555-LG-Muenchen-Az-7-O-524507-Lizenzverletzung-der-GPL.html und Picot, Henriette, „Die deutsche Rechtsprechung zur GNU General Public License”, in: Open-Source-Jahrbuch 2008, S. 184 ff., 194

[87] Urteil vom 28. 09. 2010 - I-20 U 41/09 xt:Commerce; siehe auch Van den Brande / Coughlan / Jäger a.a.O. S. 214 f.

[88] Urteil vom 10.12.2010, Az. 406 O 50/10, vgl. http://openjur.de/u/69386.html

[89] Teilurteil vom 20.1.2011, Az. I-8 O 293; das Urteil kann angesehen werden unter: http://www.telemedicus.info/urteile/Urheberrecht/Open-Source-/1148-LG-Bochum-Az-I-8-O-29309-Ansprueche-bei-Verletzung-der-LGPL.html; vgl. auch Van den Brande / Coughlan / Jäger a.a.O., S. 209

Im Jahr 2011 urteilte das Landgericht Berlin im Fall AVM gegen Cybits, dass die in der Hardware installierte GPL-Software modifiziert und wiederum installiert werden darf.[90] Allerdings durfte die konkrete, fehlerbehaftete Modifikation, die zu erheblichen Wartungsaufwänden des Hardware-Lieferanten führte, nicht weiterverbreitet werden.[91]

Im Jahr 2012 urteilte das Oberlandesgericht Düsseldorf im Fall Enigma, dass der markenrechtlich geschützte Name einer GPL-Software im Sinne eines Werktitels verwendet werden darf, solange die Pflichten der GPL erfüllt sind.[92] Der Senat stellte überdies fest, dass Software unter GPL zusammen mit einer kostenpflichtigen Hardware vertrieben werden darf.[93]

Im Jahr 2013 stellte das Landgericht Hamburg im Fall H.Welte gegen Fantec fest, dass für eine Software unter GPL die aktuell verwendete Version des Quellcodes bereitgestellt werden muss.[94] Auch wenn die Software-Erstellung im Auftrag durch einen Dritten erfolgt, ist dennoch der Auftraggeber für die rechtliche Richtigkeit aller Angaben verantwortlich.[95] Um seiner Sorgfaltspflicht nachzukommen, muss er jedenfalls auf entsprechenden Hinweis geeignete Werkzeuge zur Code-Analyse einsetzen oder Fremdwissen einkaufen.

Im Juli 2015 entschied das Landgericht Halle/Saale, dass die Heilungsmechanismen der GPL v3.0 bei einer erstmaligen Verletzung der Lizenz nicht rechtliche Maßnahmen zur Vorbeugung weiterer Verletzungen ausschließen[96]. Somit hat der Lizenzgeber auch bei einer Erstverletzung der Lizenz einen Anspruch auf eine strafbewehrte Unterlassungserklärung.

Auch aus **Frankreich** liegen bereits mehrere Urteile zur Einhaltung von Open-Source-Lizenzen vor[97]. Im Jahr 2003 urteilte der TGI Paris im Fall Mandrake gegen Logidee, dass Logidee als Ersteller („made by“) in den im Auftrag von Mandrake erstellten Manualen genannt werden muss, da diese unter die GFDL und GPL gestellt worden waren.[98]

Im Jahr 2007 erklärte der TGI Paris im Fall der Firma Educaffix gegen eine öffentliche Forschungsinstitution[99] den Vertrag zum Verkauf einer Lernsoftware für nichtig, da die öffentliche Institution den Aufwand für den Ersatz der Komponente JatLite unter GPL erheblich unterschätzt hatte, und somit die wirtschaftliche Nutzbarkeit nicht entsprechend möglich war[100].

Im Jahr 2009 hob der Court d“Appel de Paris einen Vertrag zwischen der öffentlichen Hand und der Firma EDU4 auf, da EDU4 in die bereitgestellte Software auch Komponenten unter GPL integriert hatte, diese Nutzung aber nicht deklariert und auch keine der Verpflichtungen aus der GPL erfüllt hatte. Sogar die Copyrights der GPL-Software waren entfernt worden. Das Gericht stellte fest, dass

---

[90] Urteil vom 08.11.2011, Az. 16 O 255/10 (http://fsfe.org/activities/ftf/lg-urteil-20111118.pdf); vgl. auch Laurent a.a.O., S. 10

[91] Vgl. http://www.heise.de/open/meldung/AVM-vs-Cybits-Gericht-staerkt-GPL-1389738.html und Van den Brande / Coughlan / Jäger a.a.O., S. 210 ff.

[92] Urteil vom 24.04.2012, Az. I-20 U 176/11 , vgl. auch Van den Brande / Coughlan / Jäger a.a.O., S. 215 ff.

[93] http://www.damm-legal.de/olg-duesseldorf-modifikationen-einer-Open-Source-Software-duerfen-so-lange-unter-dem-urspruenglichen-namen-der-software-vertrieben-werden-wie-sie-ihr-aehnlich-bleibt

[94] Urteil vom 14.06.2013, Az. 308 O 10/13 (http://www.heise.de/open/meldung/GPL-Urteil-Quellen-und-Binaries-muessen-gleiche-Version-haben-1897161.html); vgl. auch Van den Brande / Coughlan / Jäger a.a.O, S. 212 ff.

[95] Urteil des LG Hamburg vom 14.06.2013, Az. 308 O 10/13 unter Entscheidungsgründe A.I.1.b.)

[96] Urteil vom 27.7.2015, Az. 4 O 133/15, http://www.landesrecht.sachsen-anhalt.de/jportal/?quelle=jlink&docid=JURE150012453&psml=bssahprod.psml&max=true

[97] Vgl. Laurent a.a.O., S. 11 ff.; Van den Brande / Coughlan / Jäger a.a.O, S. 152 ff. und Bernard Lamon, “Le droit des licences Open Source”, 2009 (http://www.bernardlamon.fr/wp-content/uploads/2009/07/livre-blanc-v3-aout-2009.pdf), S. 55 ff.

[98] TGI Paris, Urteil vom 25.02.2003, Laurent a.a.O., S. 11

[99] Le Centre national de la recherche scientifique, Université Joseph Fournier

[100] Vgl. TGI Paris, Urteil vom 28.03.2007, Laurent a.a.O., S. 12; siehe auch Lamon a.a.O., S. 55 ff.

auch auf die rechtliche Integrität der Software zu achten ist[101]. Bemerkenswert ist, dass der Kläger ein Nutzer der Software und nicht der Urheberrechtsinhaber war.[102]

Zu Verstößen gegen **die Creative Common Lizenzen** gibt es in einigen **europäischen Ländern** Urteile sowohl bei Verwendung von Bildern als auch von Musik[103]. In Deutschland verurteilte im Jahr 2010 das Landgericht Berlin im Fall Deutsche Volksunion gegen Nina Gerlach erstere wegen Nicht-Beachtung der „CC Attribution – ShareAlike 3.0 Unported"-Lizenz bei der Verwendung eines Fotos.[104] Die Deutsche Volksunion hatte weder den Namen der Fotografin genannt, noch auf den Lizenztext verwiesen.

Das Oberlandesgericht Köln entschied im Jahr 2014 im Fall gegen den öffentlichen Rundfunksender Deutschlandradio, dass sich die Frage, ob eine kommerzielle Nutzung eines Fotos bei Verwendung auf der Website vorliegt oder nicht, aus der zugrundeliegenden Creative Commons Attribution NonCommercial 2.0"-Lizenz im konkreten Fall nicht eindeutig beantworten lässt. Daher wird die Unklarheitenregelung zugunsten des Beklagten angewandt und der Tatbestand der kommerziellen Nutzung verneint.[105] Das Entfernen des Autorennamens bei der Beschneidung des Bildes stellt jedoch eine Verletzung der Lizenz dar, die zum Wegfall der Nutzungsrechte führt. Anders als das LG Bochum im Teilurteil vom 20.1.2011, Az. I-8 O 293 (vgl. oben) für GPL 2.0 kommt das OLG Köln zu dem Ergebnis, dass bei einer Verletzung der Creative Commons 2.0 BY NC unported kein finanzieller Schadensersatz zu leisten ist.

In den **Vereinigten Staaten von Amerika** wurde die Durchsetzbarkeit von Open-Source-Lizenzen durch mehrere Gerichtsurteile bestätigt.

Im Jahr 2002 wurde die rechtliche Wirksamkeit der GPL im Fall MySQL AB gegen Progress Software Corp und ihre Tochterfirma Nusphere indirekt anerkannt.[106]. Der Streit wurde mit einem Vergleich beendet.

Im Fall Wallace gegen IBM wurde im Jahr 2006 ein Verstoß gegen das Wettbewerbsgesetz bei Bereitstellung von Software unter GPL verneint.[107].

Die Urheber der Software BusyBox verklagten in den Jahren 2007 / 2008 mehrere Nutzer der GPL-Software. Es kam jeweils zu einer außergerichtlichen Einigung, bei der die Beklagten den Quellcode offenlegen, einen Open-Source-Compliance-Beauftragten bestellen und die Zahlung einer unbekannten Summe an die Urheber übernehmen mussten. Das Software Freedom Law Center vertrat die Urheber vor Gericht.[108]

---

[101] Cour d'appel de Paris, Urteil vom 16.09.2009 (http://fsffrance.org/news/arret-ca-paris-16.09.2009.pdf); Laurent a.a.O., S. 14

[102] http://fsffrance.org/news/article2009-09-22.en.html

[103] Laurent a.a.O., S. 19

[104] Urteil vom 08.10.2010, Az. 16 O 458/10; Laurent a.a.O., S. 28

[105] https://netzpolitik.org/wp-upload/OLG-K%C3%B6ln-CC-NC-Entscheidung.pdf und http://medien-internet-und-recht.de/volltext.php?mir_dok_id=2656

[106] Entscheidung des U.S. District Court of Massachussetts vom 28.02.2002 (https://www.gnu.org/press/2002-03-01-pi-MySQL.html und (http://corporate.findlaw.com/intellectual-property/court-evaluates-meaning-of-derivative-work-in-an-open-source.html); vgl. auch Lamon a.a.O., S. 52 f.

[107] Urteil in zweiter Instanz vom 09.09.2006: http://caselaw.findlaw.com/us-7th-circuit/1162366.html; siehe auch Van den Brande / Coughlan / Jäger a.a.O, S. 691 und Lamon a.a.O., S. 53 f.

[108] Vgl. Klageschrift unter: http://www.softwarefreedom.org/resources/2009/busybox-complaint-2009-12-14.pdf und http://www.softwarefreedom.org/news/2009/dec/14/busybox-gpl-lawsuit/

Im Jahr 2008 urteilte der Federal Court im Fall Jacobsen gegen Katzer, dass Open-Source-Lizenzen gerichtlich durchsetzbar sind und erließ somit ein bedeutendes Grundsatzurteil. Im konkreten Fall handelte es sich um die Artistic License und die Pflicht, Änderungen kenntlich zu machen.[109]

Im selben Jahr erhob die Free Software Foundation erstmals Klage wegen Verletzung der GPL. Verklagt wurde Cisco wegen unrechtmäßiger Nutzung in mehreren Produkten, die die aufgekaufte Firma Linksys in den Konzern gebracht hatte. Die gerichtliche Einigung im Jahr 2009 erforderte die Offenlegung des Quellcodes und die Zahlung einer unbekannten Summe an die FSF.[110]

Im Jahr 2010 wurde WestingHouse verurteilt, wegen Verstoßes gegen die GPL beim Vertrieb der Software Busybox den Verkauf der betroffenen Produkte – Fernseher - einzustellen und zusätzlich einen Schadensersatz von 90.000 Dollar und die Prozesskosten von ca.47.000 Dollar zu zahlen.[111]

Ein mehrjähriger, teilweise noch immer nicht abgeschlossener Rechtsstreit spiegelte auch die Auseinandersetzung zwischen kommerzieller Software-Industrie und der stärker werdenden Open-Source- Bewegung wider. Von der Firma SCO wurde im Jahr 2003 die Firma IBM wegen unrechtmäßiger Nutzung von angeblich unter SCO-Urheberrecht stehendem Unix-Code in Linux verklagt, welches unter OSS-Lizenzen bereitgestellt wurde.[112] In der Folge wurden von SCO weitere Linux-Distributoren verklagt. In das Umfeld gehören auch die Streitfälle zwischen Novell und SCO, sowie Novell und Microsoft. Im Jahr 2010 entschied der Utah U.S. District Court, dass Novell die Urheberrechte an dem strittigen Linux-Code besitzt, und SCO daher keine Ansprüche geltend machen kann.[113] Der Fall, oder eigentlich die Fälle, erregten viel Aufsehen, und wurden in vorbildlicher Weise für die Öffentlichkeit auf http://www.groklaw.net/ einschließlich Verweisen auf sämtliche Rechtsunterlagen, wie Klagen und Urteile, dokumentiert.

Im Jahr 2014 klärte der Federal Court im Fall Ameriprise gegen Versata nicht, ob auch der Nutzer einer Software unter GPL die Einhaltung der Pflichten des Lieferanten einklagen kann,[114] erlaubte jedoch eine weitere Bearbeitung des Falles unter „statelaw“. Eingeklagt wird die Offenlegung der Quellen eines abgeleiteten Werks.[115]

XimpleWare, dessen Software von Versata / Ameriprise und deren Kunden genutzt wurde, verklagte diese Firmen wegen Patentverletzungen. Das Urteil stellte fest, dass die GPL v2.0 im Falle des bloßen Ablaufens der Software eine implizite Patentlizenz einräumt, und diese gültig ist, solange nicht die GPL v2.0 verletzt wird.[116]

Kommentiert [KT25]: Kommentar Frau Schnizer: Die Frage der impliziten Patentlizenz im Falle der Weitergabe der Software unter GPL v2.0 wurde nicht weiter behandelt, und ist somit noch offen.

In **Korea** wurde im Jahr 2006 im Fall Elimnet gegen Haionnet die Rechte und Pflichten aus der GPL bei der Urteilsfindung zugunsten des Geschäftsgeheimnisses vollständig ausgeblendet. Dies gibt - Anlass zur Sorge, ob mittelfristig Open-Source- Lizenzen im koreanischen Rechtssystem anerkannt werden.[117]

In **Israel** wurde im Jahr 2011 die widerrechtliche kommerzielle Nutzung von Fotos unter der nicht-kommerziellen Form der Commons Creative Lizenz sowie die fehlende Attributierung bestraft, wobei

---

[109] United States Court of Appeals, Federal Circuit, No. 2008-1001, Aug. 13, 2008 (http://caselaw.findlaw.com/us-federal-circuit/1189790.html); vgl. auch Van den Brande / Coughlan / Jäger a.a.O, S. 689

[110] https://www.fsf.org/news/2009-05-cisco-settlement.html und http://www.heise.de/open/meldung/Free-Software-Foundation-und-Cisco-einigen-sich-219823.html

[111] http://www.linuxplanet.com/linuxplanet/reports/7145/1 und http://www.groklaw.net/articlebasic.php?story=20100803132055210

[112] http://www.groklaw.net/staticpages/index.php?page=20031016162215566

[113] http://groklaw.net/pdf2/Novell-878.pdf

[114] http://www.ifross.org/node/1542 und http://opensource.com/law/14/7/lawsuit-threatens-break-new-ground-gpl-and-software-licensing-issues

[115] http://www.gpo.gov/fdsys/granule/USCOURTS-txwd-1_14-cv-00012/USCOURTS-txwd-1_14-cv-00012-0

[116] http://opensource.com/law/14/12/gplv2-court-decisions-versata

[117] Van den Brande / Coughlan / Jäger a.a.O, S. 304

ein Strafgeld für jedes einzelne Foto,[118] ein Strafgeldzuschlag für Nutzung in einem Buch und die Übernahme der Prozessverfahrenskosten verlangt wurde.

Es bleibt anzumerken, dass bisher weltweit kein Urteil bekannt ist, in dem die Frage nach dem Vorliegen eines abgeleiteten Werkes eingehender behandelt wurde. Allerdings zeigte sich diese Frage indirekt in den oben erwähnten Fällen MySQL, 2002[119], Educaffix, 2007[120], und Versata/Ameriprise 2014.

6.13 Empfehlungen zur rechtskonformen Anwendung von Open-Source-Software

Kommentiert [KT26]: Kommentar Herr Teichert: 6.13 und 7.2.2 könnten m.E. noch stärker harmonisiert werden.

6.13.1 Rechtliche Risiken bei Open-Source-Software

Wie bereits erwähnt, ist Open-Source-Software zwar kostenlos, aber nicht frei von lizenzrechtlichen Vorgaben erhältlich. Die unkontrollierte Verwendung von Open-Source-Software in einem Unternehmen und in seinen Produkten führt zu nicht kalkulierbaren rechtlichen Risiken sowohl im Unternehmen selbst als auch in der nachfolgenden Lieferkette. Realisieren sich solche rechtlichen Risiken, kann sich der Kostenvorteil einer unentgeltlichen OSS-Verwendung schnell in sein Gegenteil verkehren.

So können Verstöße gegen Lizenzbedingungen den Verlust von Nutzungsrechten und Schadensersatzansprüche wegen Urheberrechtsverletzung nach sich ziehen. Die nicht gestattete Verwendung von Namen oder Marken eines OSS-Urhebers durch einen anderen Software-Ersteller kann zu Schadensersatzansprüchen wegen Markenrechtsverletzung führen.

Des Weiteren können Open-Source-Lizenzbedingungen für ein Unternehmen die Pflicht begründen, den Source Code aus den von ihm vertriebenen Produkten offen zu legen. Bei Mängeln der im Produkt integrierten Open-Source-Software kann das Unternehmen vom Kunden in Haftung genommen werden, wenn beim Vertrieb keine transparente Abgrenzung von verwendeter Open-Source-Software und eigener Software vorgenommen wurde. Bei einer transparenten Abgrenzung und Durchleitung der Lizenzbedingungen der Open-Source-Software an die Kunden kann dann wiederum die Wirksamkeit einer solchen Durchleitung in Frage stehen.

Ein Regress des Unternehmens gegenüber den Autoren einer Open-Source-Software kann daran scheitern, dass der Anspruchsgegner nicht eindeutig bestimmbar ist oder dessen Haftung wirksam ausgeschlossen wurde.

Weitere rechtliche Risiken:

- Gewährleistungsrisiken, wenn OSS in der Lieferkette weitergegeben wird, ausreichender Support der Urheber für Fehlerbehebung oder Aktualisierung der Software aber nicht durchgesetzt werden kann
- Nur eingeschränkte Verwertbarkeit eigener Software, in die OSS-Komponenten integriert sind
- Wer Open-Source-Komponenten in andere Produkte integriert, unterliegt bei Fehlerhaftigkeit der Open-Source-Komponenten Produzentenhaftung und kann bei Verstößen gegen Verkehrssicherungspflichten haftbar sein
- Der Verstoß gegen Lizenzbedingungen kann dazu führen, dass die Befugnis zur Verbreitung einer Open-Source-Software entzogen wird. Dies wäre vor allem dann problematisch, wenn der Lizenzverletzer die Software in eigenen Produkten integriert hat
- Copyleft-Effekt: eigene Änderungen einer Software müssen öffentlich gemacht werden
- Patentleft-Effekt?

Kommentiert [KT27]: Hier zu erwähnen? Wenn ja, Risiko noch etwas detaillierter darstellen, da sonst nicht verständlich!

[118] http://www.law.co.il/en/news/israeli_internet_law_update/2011/01/18/Israeli-court-enforces-a-creative-commons-license
[119] http://corporate.findlaw.com/intellectual-property/court-evaluates-meaning-of-derivative-work-in-an-open-source.html
[120] http://www.crid.be/pdf/public/5845.pdf

6.13.2 Management von Open-Source-Software

Um den rechtlichen Risiken von Open-Source-Software aus dem Weg zu gehen, sollte ein Management der im Unternehmen vorhandenen und genutzten Open-Source-Software eingerichtet werden.

Kommentiert [KT28]: Ausführungen zum Lizenzmanagement enthält auch Kapitel 7.3. Soll diese Dopplung erhalten bleiben?

6.13.2.1 Erfassung der verwendeten Open-Source-Software

Dazu sind zunächst die **im Unternehmen vorhandenen und verwendeten OSS-Komponenten** einschließlich ihrer Autoren bzw. Bezugsquellen zu **identifizieren**, zu **erfassen** und mit zugehörigen Lizenztexten zu **dokumentieren**. Dies ist besonders wichtig für OSS-Komponenten, die in eigenen Produkten des Unternehmens Verwendung finden. Für die Erfassung können entsprechende Tools zur Unterstützung herangezogen werden. Die Implementierung geeigneter Verfahren für die Steuerung des Einsatzes von Open Source Software gehören zu den Organisationspflichten im Unternehmen, ihr Fehlen kann (z.B. aufgrund sog. Organisationsverschulden) zu persönlicher Haftung führen. Weitere Einzelheiten zu Erfassung und Verwaltung von OSS-Lizenzen sind im Kapitel 7.3 (Lizenzmanagement und Compliance) dargestellt. Daneben sollte eine Befragung der relevanten Mitarbeiter im Unternehmen nach bereits eingesetzter Open-Source-Software stattfinden.

Auch im Release-Prozess für eigene Software-Produkte sollten Schritte zur Identifizierung von OSS-Komponenten eingerichtet werden, sodass bei der abschließenden Freigabe eines Produkts durch eine letzte Prüfungsinstanz klar ist, welche OSS-Komponenten in einem Produkt enthalten sind und welche Risiken dies birgt.

Abschließend sei darauf hingewiesen, dass das Management von OSS-Komponenten und OSS-Lizenzen im Unternehmen als kontinuierlicher Prozess anzusehen ist und nicht als einmaliges und abschließbares Projekt.

6.13.2.2 Unternehmensinterne Regeln zur Verwendung von Open-Source-Software

Kommentiert [KT29]: Aus meiner Sicht wäre es sinnvoll, diesen Abschnitt in Kapitel 7.3 zu überführen und damit im Zusammenhang mit Compliance darzustellen.

Zum OSS-Management gehört es nicht zuletzt, die eigenen Software-Entwickler im Unternehmen für die OSS-Problematik zu sensibilisieren. Ihnen müssen Richtlinien an die Hand gegeben werden, z.B. hinsichtlich Lizenzkompatibilitäten. Dabei ist zu berücksichtigen, ob das Unternehmen unveränderte oder veränderte Open-Source-Software verwendet und ob diese nur zur internen Verwendung oder auch zur externen Verwendung dient. Der Vertrieb einer Open-Source-Software in modifizierter Form stellt dabei im Allgemeinen das größere und die rein interne Verwendung einer unveränderten Open-Source-Software das geringere Risiko dar. Das Risiko erhöht sich tendenziell, wenn die verwendete Software unter einer Lizenz mit Copyleft-Effekt steht.

In der Praxis haben sich folgende Vorgaben für interne OSS-Richtlinien als sinnvoll erwiesen:

- Kurze und verständliche Beschreibung der Rahmenbedingungen zum Einsatz von Open Source Software (Tools, Bibliotheken und Server-Software)
- Pflicht zur Einholung der Einwilligung einer zuständigen internen Stelle für die Verwendung, Bearbeitung oder Distribution der jeweiligen Open Source Software.
- Einführung einer unternehmensinternen Genehmigungspflicht für die Bearbeitung und Verbindung von Open Source Software mit eigener Software, deren Weitergabe beabsichtigt ist.

  Für die effektive Umsetzung der internen Regelungen empfiehlt sich eine Formalisierung der Einwilligungsprozesse für die Verwendung von Open Source Software. Um dabei eine unnötige Zentralisierung oder Bürokratisierung zu vermeiden, sollte das Maß an Risiko, welches das Unternehmen im Hinblick auf die Nutzung von Open Source Software bereit ist einzugehen, zur Bestimmung der Grenzen der Formalisierung dienen.
- Einholung von Bescheinigungen oder Garantien über Bestandteile von Open Source Software bei der Beschaffung von Software
- ggf. den unternehmensinternen Einsatz oder bestimmte Verwendungsszenarien von Software mit unliebsamen Lizenzkonsequenzen untersagen

Kommentiert [KT30]: Meinen diese beiden Bullit-Points nicht dasselbe?

Kommentiert [KT31]: Welche Bescheinigungen / Garantien sind hier genau gemeint?

- OSS-Komponenten und proprietäre Software streng auseinanderhalten und getrennt lizenzieren; auch in den Produktbeschreibungen sollte eine transparente Abgrenzung von Open-Source-Software und selbst erstellten Software-Komponenten vorgenommen werden
- Pflicht zur Verwendung bestimmter Allgemeiner Geschäftsbedingungen (AGB) oder Vereinbarung individueller Regeln beim Vertrieb oder Beschaffung von Software, die Open Source Software enthält
- Lizenztexte für verwendete Open-Source-Komponenten lesen und befolgen
- Lizenzbedingungen für OSS-Komponenten, die in den Produkten des Unternehmens verwendet werden, sind den Kunden des Unternehmens zugänglich zu machen (am besten vorab)
- Für den Fall, dass Rechte Dritter an der Open Source Software die vertragsgerechte Leistungserbringung des Anbieters unmöglich machen, sollte sich der Anbieter im Vertrag über seine Leistung ein Notausstiegsrecht vorbehalten. Diese entsprechende vertragliche Formulierung könnte wie folgt lauten: „Werden durch eine Leistung des Anbieters Rechte Dritter verletzt, wird der Anbieter nach eigener Wahl und auf eigene Kosten a) dem Kunden das Recht zur Nutzung der Leistung verschaffen oder b) die Leistung rechtsverletzungsfrei gestalten oder c) die Leistung unter Erstattung der dafür vom Kunden geleisteten Vergütung (abzüglich einer angemessenen Nutzungsentschädigung) zurücknehmen, wenn der Anbieter keine andere Abhilfe mit angemessenem Aufwand erzielen kann."
- Vorgaben für die Interaktion mit Open-Source-Software-Gemeinschaften, deren Werke genutzt werden
- regelmäßige Aktualisierung der internen Open-Source-Software-Richtlinien anhand sich verändernder Open-Source-Lizenzbedingungen und geänderter Verwendungsszenarien.

Alle Vertragsmuster und laufenden Verträge sollten auf die Einhaltung der unternehmensintern festgelegten Regeln überprüft werden.
Interne Richtlinien können nur erfolgreich sein, wenn die davon betroffenen Unternehmensangehörigen darüber informiert und in ihrer Befolgung sowie im Umgang mit den geeigneten Mitteln geschult werden.

Kommentiert [KT32]: Sinnvolle Vorgabe für eine interne OSS-Richtlinie? Diese Empfehlung richtet sich doch eigentlich an die Rechts- oder Vertriebsabteilung. Noch konkreter fassen: welche AGB oder individuellen Regelungen sollen genau vereinbart werden und zu welchem Zweck?

Kommentiert [KT33]: Kommentar Herr Teichert: Laut 6.7.1: „am besten vorab zugänglich"

6.14 Zusammenfassung

Open-Source-Software befindet sich nicht im „rechtsfreien Raum". Ihre Nutzung unterliegt Beschränkungen und begründet Verpflichtungen, auch wenn dafür kein Entgelt zu zahlen ist.

Die unkontrollierte Nutzung von Open Source Software in einem Unternehmen und in dessen Produkten kann ein nicht kalkulierbares rechtliches und finanzielles Risiko sowohl für das Unternehmen als auch für die nachfolgende Lieferkette darstellen. So kann beispielsweise für das Unternehmen die Pflicht bestehen, den Source Code des vertriebenen Produktes offen zu legen. Bei Mängeln der im Produkt integrierten Open-Source-Software kann das Unternehmen vom Kunden in Haftung genommen werden, wenn beim Vertrieb keine transparente Abgrenzung von verwendeter Open-Source-Software und selbst erstellter Closed-Software vorgenommen wurde. Ein Regress des Unternehmens gegenüber dem Autor der Open Source Software kann daran scheitern, dass der Anspruchsgegner nicht eindeutig bestimmbar ist oder dessen Haftung wirksam ausgeschlossen wurde.

Einige wesentliche Hinweise:

- Für Open Source Software gelten dieselben Rechtsvorschriften wie für andere Software auch.
- Rechtliche Regelungen und Beschränkungen für Open Source Software werden insbesondere in deren Lizenzbedingungen getroffen.
- Für die Lizenzbedingungen von Open Source Software kann ausländisches Recht gelten.
- Im Unternehmen ist die Erfassung von Open-Source-Software sowie eine Steuerung und Kontrolle ihrer Verwendung notwendig.
- In Unternehmen verwendete Open-Source-Software sollte erfasst und ihre Verwendung gesteuert und kontrolliert werden. Die Implementierung geeigneter Verfahren hierfür gehört in den Bereich der Organisationspflichten im Unternehmen, ihr Fehlen kann (z.B. aufgrund sog.

Organisationsverschuldens) zu persönlicher Haftung führen. Die Steuerung des Einsatzes von Open Source Software erfordert zumindest:

- technisches Management und
- rechtliches Lizenzmanagement

für die verwendete Open-Source-Software.

- Der Vertrieb von Open-Source-Software oder von Open-Source-Software zusammen mit anderer Software führt zu Haftungsrisiken.
- Der Vertrieb veränderter Open Source Software kann zur Offenlegung der geänderten Source-Codes verpflichten.
- Auch der Vertrieb von Open Source Software zusammen mit selbst erstellter Software kann zur Offenlegung der geänderten Source-Codes verpflichten.
-
- Bei der Gestaltung von Verträgen über Leistungen im Zusammenhang mit Open Source Software (etwa Implementierung, Anpassung, Zusatzprogrammierung etc.) sollten fachkundige unternehmensinterne oder externe Rechtsberater beteiligt werden.

Diese Darstellung kann angesichts der Vielgestaltigkeit von Open Source Software und ihrer Verwendung weder Anspruch auf detaillierte Betrachtung einzelner Einsatzszenarien noch auf Vollständigkeit erheben. Sie soll vielmehr die auch im Zusammenhang mit notwendigem Risk-Management erforderliche Sensibilität für den Umgang mit Open Source Software fördern.

7. Open-Source-Software im Unternehmen

Noch vor 10 Jahren galt der Einsatz von Open-Source-Software in Unternehmen als revolutionär. Firmen, die öffentlich erklärten, dass sie Open-Source- Software einsetzen, wurden argwöhnisch und misstrauisch beobachtet. Heute hingegen zeigen alle Studien zum Thema einheitlich, dass der Einsatz von Open-Source-Software in Unternehmen ganz normal geworden ist. Es gibt vermutlich kein Unternehmen mehr, das keine Open Source-Software einsetzt. Umso erstaunlicher ist es, dass Open-Source-Software in vielen Unternehmen immer noch eher auf technische und rechtliche Belange reduziert wird und nicht als umfassendes neues Produktions-, Vertriebs- und Geschäftsmodell betrachtet wird. Dass dies der Bedeutung von Open-Source-Software bei Weitem nicht gerecht wird, zeigt die Tatsache, dass Open-Source-Software seit einigen Jahren die ITK-Ökosysteme revolutioniert. Dies geschieht oftmals leise und am Anfang unbemerkt – die Auswirkungen dieser Revolution sind trotzdem weitreichend und unübersehbar. So strukturierte sich zum Beispiel Broadcom als Unternehmen völlig neu, um mit den Implikationen von Open-Source-Software besser zurechtzukommen. Der beispiellose Siegeszug von Android, das Nokia als den dominierenden Handyhersteller förmlich in die Bedeutungslosigkeit katapultierte, zeigt eindrücklich, dass Unternehmen alle Einflussfaktoren von OSS berücksichtigen sollten und regelmäßig überprüfen müssen, ob ihre Strategie noch geschäftsfördernd ist oder nicht. Hier wird es schwierig für viele Unternehmen, da sie keine Strategie für den Umgang mit Open-Source-Software haben.

7.1 OSS-Strategieentwicklung im Unternehmen

Die folgenden Kapitel sollen Unternehmen die Definition einer Open-Source-Strategie erleichtern und eine Hilfestellung bei der Ergreifung der notwendigen flankierenden Maßnahmen bieten. Ob alle genannten Betrachtungen, Analysen und Stakeholder in den Unternehmen durchgeführt werden oder vorhanden sind, hängt dabei sehr von der Größe des Unternehmens ab.

Ausgangspunkt der Betrachtung mag eine gängige Definition des Begriffs „Strategie“ sein:

> „Unter Strategie werden in der Wirtschaft klassisch die (meist langfristig) geplanten Verhaltensweisen der Unternehmen zur Erreichung ihrer Ziele verstanden. In diesem Sinne zeigt die Unternehmensstrategie in der Unternehmensführung, auf welche Art ein mittelfristiges (ca. 2–4 Jahre) oder langfristiges (ca. 4–8 Jahre) Unternehmensziel erreicht werden soll. ... Im Zusammenhang mit der Unternehmensstrategie wird oft von den vorgeordneten Konzepten der Vision und des Unternehmensleitbildes gesprochen, sowie von Strategischem Management. Als nachgeordnet werden Teilstrategien (Marketingstrategie, Finanzierungsstrategie etc.) und die taktische (mittelfristige) sowie die operationale (kurzfristige) Ebene angesehen.“[121]

Bezogen auf den Einsatz von Open-Source-Software ist zu bestimmen, in wieweit OSS im Sinne der Geschäftsziele eingesetzt werden kann. Zur Ableitung einer solchen Strategie gilt es verschiedene Aspekte zu beachten.

7.1.1 Randbedingungen und Einflussfaktoren

Bevor die eigentliche Strategieentwicklung starten kann, müssen Einflussfaktoren und Randbedingungen festgehalten und analysiert werden, die die Entwicklung der Strategie beeinflussen. Hierzu zählen:

- Markt, in dem die Firma agiert
- Konkurrenten
- Kunden / Kundenstruktur
- Kunden der Kunden (sofern relevant).

[121] Vgl. die allgemeine Begriffserklärung in Wikipedia (http://de.wikipedia.org/wiki/Strategie_(Wirtschaft))

7.1.2 Strategieentwicklung

Die Entwicklung einer geeigneten OSS-Strategie ist weder Sache der Entwicklungsabteilungen noch der Rechtsabteilungen. Planung und vorab definierte Ziele sind bedeutende Bausteine – Strategien entstehen nicht zufällig und können auch nicht von einzelnen Abteilungen eines Unternehmens formuliert werden. Open-Source-Software und der Umgang damit betrifft Unternehmen als Ganzes. Deswegen ist die Entwicklung einer geeigneten OSS-Strategie Aufgabe des obersten Managements und unter Einbeziehung aller Stakeholder durchzuführen. Je nach Größe der Firma sind folgende Stakeholder zu involvieren:

- Entwicklungsabteilungen
- Einkauf
- Rechtsabteilung (inklusive Patent- und Markenrechtsabteilung)
- Personalabteilung
- Service
- Vertrieb
- IT-Infrastruktur
- Produktmanager / Portfoliomanager
- Merger und Akquisition-Abteilung
- Finanzabteilung / ggf. Investoren
- Kommunikation / Marketingabteilung
- Marktanalyse-Abteilung (sofern vorhanden)
- Technologie- und Zukunfts-Abteilung (Strategie-Abteilung, sofern vorhanden)

Dass die Formulierung einer geeigneten Open-Source-Strategie Aufgabe des obersten Managements ist und alle Stakeholder zu involvieren sind, gilt auch, wenn die Open-Source-Strategie nur eine „Supporting Strategy“ ist und die Primärstrategie etwa lautet:

- „Gewinnung von Marktanteilen durch Reduktion der Herstellkosten“
- „Erhöhung der durch die Produkte und Services zu erzielende Marge“
- „Verteidigung des eines bestimmten Marktes oder Marktsegmentes gegen Konkurrenten“.

Solche oder ähnliche Strategien sind häufig in der traditionellen Industrie anzutreffen. Open-Source-Software kann unter anderem ein Mittel sein, um die Primärstrategie umzusetzen. Allerdings geht das nur, wenn man eine Open-Source-Strategie definiert und diese auch konsequent umsetzt. Folgende Fragen sollen die Herleitung und Formulierung einer geeigneten Open-Source- Strategie unterstützen:

- Wollen wir Open-Source-Software einsetzen?
- Welches Ziel wollen wir erreichen?
- Wie ist die Haltung / Einstellung des Marktes, in dem wir tätig sind, bezüglich OSS?
- Was ist die Meinung unserer Leitkunden bezüglich OSS?
- Wie ist das Verhalten unserer Hauptkonkurrenten bezüglich OSS?
- Benötigen unsere Produkte besondere Zulassungen (TÜV, FDA, etc.)?
- Gibt es Start-up-Unternehmen, die Marktanteile gewinnen?
- Gibt es OSS-Projekte, die Teile unseres Marktes bedienen können?
- Gibt es OSS-Projekte, die für das eigene Produktportfolio enorm wichtig sind?
- Inwieweit wollen/müssen wir ein aktiver Teil des OSS Ökosystems werden?
- Welches sind die mit dem Einsatz von OSS verbundenen Risiken?
- Was muss auf jeden Fall sichergestellt werden?
- Welche Risiken sind wir bereit einzugehen?
- Wieviel Zeit und Geld wird benötigt, um die geeignete Strategie umzusetzen (welche Rollen müssen definiert werden; welche Prozess und Prozessschritte müssen definiert und eingeführt werden; muss ggf. eine Restrukturierung erfolgen etc.).

Die Antworten auf diese Fragen sind sehr stark von dem Markt abhängig, in dem das Unternehmen agiert und können aus diesem Grund in diesem Leitfaden nicht erschöpfend beantwortet werden.

7.1.3 Strategiebeispiele für die den Einsatz von OSS

Hier einige Denkanstöße für die eigene Strategieentwicklung:

Strategiebeispiel A: Die Verwendung von Open-Source-Software ist im Unternehmen nicht erlaubt

Offenkundige Vorteile:

- Abschottung gegenüber unbekannten Vertriebs-, Kooperations-, und Lizenzmodellen

Offenkundige Nachteile:

- Keine Nutzung von frei verfügbaren Gütern
- Implementierungs- und Kontrollkosten der Strategie können sich in der gleich hoch sein wie die Kosten einer Strategie „pro“ Open-Source-Software.

Strategiebeispiel B: Die Verwendung von Open-Source-Software, die unter einer strengen Copyleft-Lizenz steht, ist nicht erlaubt

Offenkundige Vorteile:

- Strikte Vermeidung unfreiwilliger Veröffentlichung von Alleinstellungsmerkmalen
- Nutzung von vielen permissiv oder schwach-copyleft-lizenzierten Werken möglich

Offenkundige Nachteile:

- Nutzung vieler frei verfügbarer Güter ist ausgeschlossen: viele Open-Source-Projekte sind unter der GPL lizenziert (vgl. dazu Kapitel 6.1); als prominentestes Beispiel sei das Betriebssystem Linux genannt.
- Die Erfüllung der Bedingungen der anderen ‘erlaubten‘ Open-Source-Lizenzen kann u.U. komplexer sein, als man es von unbedarfter Seite zunächst erwartet. Für die GPL selbst gibt es im Internet - von lizenznahen Interpreten vorgetragene - Hilfestellungen zur Erfüllung der Bedingungen der GPL.[122] Für die anderen Lizenzen steht ein solcher Support nicht so systematisch und institutionalisiert zur Verfügung.[123] Des Weiteren besteht für die GPL durch eine Reihe von Gerichtsurteilen Rechtssicherheit bezüglich der Durchsetzbarkeit der GPL und einzelner ihrer Klauseln.
-

Strategiebeispiel C: Die Verwendung von OSS, die unter strenger Copyleft-Lizenz steht, ist nur für die Einsatzart „stand-alone und unverändert“ zugelassen.

Offenkundige Vorteile:

- Strikte Vermeidung unfreiwilliger Veröffentlichung von Alleinstellungsmerkmalen
- Nutzung von vielen permissiv oder mit schwachem Copyleft lizenzierten Werken möglich
- Der große Bereich von Software unter starker Copyleft-Lizenz wird zumindest in speziellen Nutzungsszenarien zugänglich.

Offenkundige Nachteile:

- Keine Einbindung von Software unter starkem Copyleft als Komponenten in eigener Software.
- Die Erfüllung der Bedingungen anderer, erlaubter Open-Source-Lizenzen kann deutlich schwieriger sein als die Bedingungen der GPL zu erfüllen. Im Internet gibt es Hilfestellungen zur Erfüllung der Bedingungen der GPL.[124] Des Weiteren besteht durch eine Reihe von

Kommentiert [KT34]: Kommentar Herr Teichert: Ist 2.3 gemeint ?

Kommentiert [KT35]: Kommentar Herr Teichert: Meines Erachtens wäre als die zweite Stufe eher Kombinationen aus Nutzungsarten und Lizenzen / Lizenzpflichten zu betrachten.
Z.B. könnte der ungeänderte stand alone Einsatz auch von Copyleft lizenzierter Software dabei zugelassen sein.
Wer z.B. einen Linuxserver im Haus hat braucht nicht mehr über kategorisch keine GPL nachzudenken.

[122] siehe z.B. http://fsfe.org/activities/ftf/useful-tips-for-vendors.de.html
[123] Gleichwohl existieren auch für diesen Bedarf Hilfsmittel. Eine grundsätzlich gute Quelle dafür bietet etwa das OSI-Wiki an (http://wiki.opensource.org/bin/Projects/List+of+Licensing+Tools). Ein Tool, das sich speziell auch mit der Erfüllung von Nicht-GPL-Lizenzen auseinandersetzt, ist das *Open Source License Compendium* (http://dtag-dbu.github.io/oslic/)
[124] siehe z.B. http://fsfe.org/activities/ftf/useful-tips-for-vendors.de.html

Gerichtsurteilen eine gewisse Rechtssicherheit, was für andere Open-Source-Lizenzen möglicherweise nicht der Fall ist.

Strategiebeispiel D: Die Nutzung von Open-Source-Software ist erlaubt

Offenkundige Vorteile:

- Voller Zugriff auf den gesamten Pool von Open Source Software
- Volle Kooperation mit der Community ist möglich

Offenkundige Nachteile:

- Die reine Nutzung ist in den meisten Fällen keine nachhaltige Strategie
- Die Kooperation mit der Community bleibt unsystematisch
- Es besteht die Gefahr, in die „Wartungsfalle“ zu tappen: Gelegentlich wird Open-Source-Software noch modifiziert oder erweitert, bevor sie in die eigenen Produkte integriert wird. Werden diese Änderungen und Erweiterungen dann nicht an die entsprechenden Open-Source-Software-Projekte zurückgegeben, müssen sie nach einem Upgrade der Open-Source–Software-Basis erneut in den aktualisierten Softwarestand integriert werden. Das ist im Endeffekt „unproduktive“ Arbeit.

Strategiebeispiel E: Open-Source-Software ist von strategischer und wettbewerbsrelevanter Bedeutung. Wir werden ein geschätzter und aktiver Teil des Open-Source-Ökosystems und unsere Produkte und Services werden von den technischen und ökonomischen Vorteilen von Open-Source-Software profitieren.

Offenkundige Vorteile:

- Voller Zugriff auf den gesamten Pool von Open Source Software.
- Volle Kooperation mit der Community wird systematisiert auf die Ziele der Firma ausgelegt.

Offenkundige Nachteile:

- Für die Umsetzung der Strategie wird ein längerer Zeitrahmen zu veranschlagen sein, das heißt, dass es sich um ein „long term investment“ handelt. Es muss allen Beteiligten bewusst sein, das sich das Investment erst längerfristig richtig auszahlt. In diesem Zusammenhang sei noch einmal an die Definition von Strategie erinnert „ein mittelfristig oder langfristiges Unternehmensziel zu erreichen“. Demnach ist die Langfristigkeit kein Nachteil, sondern ein inhärentes Merkmal einer Strategie; das wird nur leider oft vergessen.
- Die Strategie ist nur dann Erfolg versprechend, wenn ein Umdenkprozess in den Köpfen aller Beteiligten und Verantwortlichen stattfindet und die Bereitschaft besteht, eine gewisse Lernkurve zu durchschreiten. Des Weiteren muss man sich bewusst sein, dass Teile des Unternehmens dadurch möglicherweise transparenter werden. Das ist kein Nachteil an sich, man muss es nur vorab wissen und auch damit umgehen können.

7.2 Strategiebeispiele für OSS-Anbieter

Zunächst ist zwischen der Erstellung und der Bereitstellung der Software zu unterscheiden. Im Allgemeinen geht man bei Open-Source-Software davon aus, dass sowohl die Erstellung als auch die Bereitstellung öffentlich betrieben wird. Das ist aber nicht notwendigerweise so. Streng genommen sagt der Begriff Open-Source-Software überhaupt nichts über die Art und Weise aus, wie die Software erstellt wird . Aus der Sicht des Anbieters ergeben sich verschiedene mögliche Modelle:

- Erstellung der Software wird öffentlich durchgeführt (transparent), Bereitstellung der Software der Allgemeinheit gegenüber
- Erstellung der Software wird nicht öffentlich durchgeführt (opak), Bereitstellung der Software der Allgemeinheit gegenüber

- Opake Erstellung, Bereitstellung der Software nur für die Nutzer der Software
- Transparente Erstellung, Bereitstellung der Software nur für die Nutzer.

Auch bei der Frage nach der Erstellung und Bereitstellung von Open-Source-Software gibt es verschiedene Modelle. Und wie bei der Auswahl einer Open-Source-Firmenstrategie ist entscheidend der Zweck, den man mit dem Strategiemodell erreichen will:

7.2.1 Transparente Erstellung, Bereitstellung für Allgemeinheit

Das Modell einer transparenten Erstellung und einer Öffnung für die Allgemeinheit entspricht dem weit verbreiteten Bild von Open-Source-Software. Bei dieser Vorgehensweise wird sowohl das Projektmanagement als auch die gesamte Entwicklung vollständig in der Öffentlichkeit durchgeführt. Alle Entscheidungen werden öffentlich diskutiert und getroffen. Sie sind für die Allgemeinheit nachvollziehbar, ebenso der gesamte Verlauf und der momentane Status der Entwicklung. Ermöglicht wird das durch die öffentliche Bereitstellung der Entwicklungsinfrastruktur wie Projekthomepage, Source- Code-Repositorien, Mailinglisten, Ticketsystem. Die Infrastruktur und deren Inhalte sind für alle einsehbar und nutzbar. Diese Art der Entwicklung bietet für Interessierte die niedrigste Einstiegshürde, um sich aktiv an dem Projekt zu beteiligen.

Eine öffentlich durchgeführte Entwicklung bedeutet nicht, dass jeder z.B. schreibend auf Source-Code-Repositorien oder die Projekthomepage zugreifen kann. Gut geführte Open-Source-Software-Projekte haben klare Regeln, die bestimmen, wer was darf, wie mit Beiträgen verfahren wird und wie Beiträge strukturiert sein sollen. Internetseiten wie SourceForge.net oder Github.com bieten eine komplette Entwicklungsinfrastruktur, sodass die Bereitstellung und Pflege eigener Infrastruktur entfallen kann.

7.2.2 Opake Erstellung, Bereitstellung für Allgemeinheit

Als Abstufung zur transparenten Erstellung steht die opake Entwicklung der Software durch ein oder mehrere Parteien. Sowohl das gesamte Projektmanagement als auch das Design, die Implementierung und der Test der Software werden nicht öffentlich durchgeführt. Lediglich die Resultate der Entwicklung werden unter einer Open-Source-Lizenz der Allgemeinheit zur Verfügung gestellt. Eine solche Vorgehensweise findet man des Öfteren bei öffentlich geförderten Projekten. Dort werden die sogenannten „deliverables“ oft unter einer Open-Source-Lizenz veröffentlicht, entwickelt werden sie aber häufig unter Ausschluss der Öffentlichkeit im Rahmen des Projektkonsortiums. Beiträge von Interessierten sind in der Regel auf Fragen und Fehlermeldungen beschränkt. Die Nichtnachvollziehbarkeit der Entscheidungen und die geringen Einfluss- und Gestaltungsmöglichkeiten stellen eine hohe Barriere für Interessierte dar und machen eine Beteiligung an solchen Projekten für die Meisten unattraktiv. Der Vollständigkeit halber sei erwähnt, dass es für Dritte möglich ist, einen veröffentlichten Stand abzuspalten, fortzuführen und damit ein Projekt zu starten, das unter den Gesichtspunkten „Transparente Erstellung, Bereitstellung der Allgemeinheit gegenüber“ betrieben wird.

7.2.3 Opake Erstellung, Bereitstellung nur für Software-Nutzer

Software unter einer Open-Source-Lizenz kann als ablauffähiges Programm zusammen mit dem vollständigen, zugehörigen Source Code direkt an den Benutzer der Software weitergegeben werden. Auf eine Veröffentlichung im Internet verzichtet der Anbieter. Ein derartiges Vorgehen ist sehr stark verwandt mit den bisherigen proprietären Verteilungsmodellen und schließt de facto eine Beteiligung anderer Personen an der Entwicklung sowie den Einsatz der Software durch Dritte aus.

Auf den ersten Blick hat es den Anschein, dass ein solches Modell nicht mit Open-Source-Lizenzen vereinbar ist – es ist jedoch rechtlich einwandfrei (siehe auch Kapitel 6). Die allermeisten Open-

Source- Lizenzen verlangen keine Veröffentlichung des Source Codes im Internet. Gegenteilige Annahmen sind eine Legende. Sie hat ihren Ursprung vermutlich in der Zeit der FUD-Kampagnen (FUD = Fear, Uncertainty and Doubt), als versucht wurde Open-Source-Software in ein schlechtes Licht zu rücken. Selbst die GPL-2.0 (GNU General Public License v2.0), von der häufig behauptet wird, dass sie eine „Veröffentlichung" des Source Codes fordert, lässt das hier beschriebene Konzept zu. So steht in Paragraph 3 der GPL: „3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following: a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange;…" Das hier beschriebene Modell ist somit konform zur GPL-2.0. Abschließend sei noch gesagt, dass man es nicht in der Hand hat zu bestimmen, was die Empfänger mit der Software (Programm und korrespondierendem Source Code) machen. Welche Rechte der Empfänger an der Software erwirbt, ist, um in dem Beispiel der GPL-2.0 zu bleiben, in der GPL-2.0 geregelt. Diese Rechte dürfen auch nicht eingeschränkt werden (Paragraph 6 der GPL-2.0).

7.3 Lizenzmanagement und Compliance

Unabhängig davon, welche der oben genannten Varianten gewählt wird: Wenn bereits existierende bzw. externe Open-Source-Software ein Teil der Lieferung oder Veröffentlichung darstellt, muss der Anbieter sicherstellen, dass die involvierten Lizenzen eingehalten werden. Grundlage hierfür bildet das Lizenz-Management, dessen wesentliche Merkmale im Folgenden erläutert werden.

Das Lizenz-Management umfasst folgende Aspekte:

1. Erfassung der verwendeten Lizenzen
2. Durchführung von Lizenzinterpretationen
3. Verwaltung von Lizenzinterpretationen
4. Definition der Umsetzungsmöglichkeiten von Lizenzinterpretationen
5. Verifikation und Erfassung der Umsetzung zur Lieferung.

7.3.1 Erfassung der verwendeten Lizenzen

Die systematische Erfassung der verwendeten externen Software und deren Lizenzen bildet die Basis aller weiteren Tätigkeiten und Maßnahmen des Lizenz-Managements. Weder die systematische Erfassung von Third-Party-Software noch das Lizenzmanagement ist spezifisch für Open-Source-Software, sondern eine allgemeine Aufgabe, die jedes Unternehmen im Rahmen der Compliance-Tätigkeiten durchzuführen hat.

Alle OSS-Komponenten, die auf Rechnern im Unternehmen installiert oder in Produkten des Unternehmens eingearbeitet sind, sollten erfasst werden. Das Ergebnis müsste kontinuierlich aktualisiert und gepflegt werden, damit auch Änderungen in der Konzeption systematisch berücksichtigt werden. Die mit Open-Source-Software befassten Mitarbeiter müssten dazu die maßgeblichen Informationen - inklusive solche zur Bestimmung und Verwendung der Open-Source-Software im Unternehmen - immer vor Beginn einer Arbeit mit der Open-Source-Software eintragen.

Eine entsprechende firmeneigene Datenbank sollte folgende Informationen beinhalten:

- Name und Version der Open-Source-Software
- Autor bzw. Bezugsquelle der Open-Source-Software
- Lizenzart und Version der Lizenz der Open-Source-Software
- Beginn der Nutzung

- Art der Nutzung
  - Programm oder Bibliothek
  - eingebettete Komponente oder eigenständige Einheit
  - in modifizierter oder in unmodifizierter Form
  - mit Weitergabe an Dritte oder ohne Weitergabe
  - in Form von Binärdateien oder als Sourcecode
- Name des mit der Nutzung der Open-Source-Software befassten Mitarbeiters
- Zielprojekt für den Einsatz der Open-Source-Software
- geplante interne Nutzung der Open-Source-Software mit oder ohne Änderungen
- geplante externe Verwendung (Kopie, Distribution) der Open-Source-Software mit oder ohne Veränderungen
- Genehmigung des OSS-Einsatzes durch den zuständigen Entscheidungsträger.

Bei kommerzieller Third-Party-Software werden die Lizenzbedingungen zwischen Lizenzgeber und Lizenznehmer – wenigstens dem Prinzip nach – ‚verhandelt'. Bei Open-Source-Software sind die Lizenzen und ihre Bedingungen vorab generell bekannt und – de facto zumeist - nicht verhandelbar. So ist es praktisch eine Sache des Lizenznehmers, also des Nutzers von Open-Source-Software, die Lizenzen zu erfassen, die für ein komplettes (Open-Source-)Software-Paket relevant sind. Das bedeutet, dass im Ernstfall – also da, wo man sich nicht auf Vorarbeiten der Community verlassen kann - das ganze Paket nach Lizenzhinweisen durchsucht werden muss. Aufgrund der Komplexität und der Größe heutiger Softwareprojekte ist eine manuelle Suche praktisch nicht durchführbar und auch nicht sinnvoll. Eine Reihe von Softwarewerkzeugen steht zur Verfügung, welche die Identifikation der in einem Software-Paket zur Anwendung kommenden Lizenzen unterstützen. Zwei dieser Softwarewerkzeuge seien in diesem Zusammenhang besonders erwähnt, da diese selbst Open-Source-Projekte sind. Es handelt sich dabei um FOSSology[125] und Ninka[126].

Jedoch ist die alleinige Identifikation der involvierten Lizenzen und der Herkunft der jeweiligen Open-Source-Pakete nicht ausreichend für den Fall, dass das Open-Source-Paket entweder in binärer Form oder im Source Code weiterverteilt werden soll, zum Beispiel als Bestandteil eines Produktes oder einer Lösung. Die meisten Open-Source-Lizenzen fordern, dass Nutzer der Software die Lizenztexte mit der Software bereitgestellt bekommen. Außerdem ist es zumindest gute Tradition, auch die Urheber der Software zu nennen. Die Art und Weise, wie und in welchem Format dies geschehen soll, ist oft nicht definiert und nur in manchen Lizenzen näher beschrieben. Einzige Bedingung ist, dass diese Informationen für Menschen lesbar und an einem leicht zu findenden Ort hinterlegt sein müssen.

[125] Vgl. FOSSology: Advancing open source analysis (URL: http://www.fossology.org/projects/fossology)
[126] Vgl. Ninka: Ninka, a license identification tool for Source Code (URL: http://ninka.turingmachine.org)

Das Fehlen eines standardisierten Formates für die Bereitstellung von Informationen für die Nutzer der Software hat bisher dazu geführt, dass viele Unternehmen unterschiedliche, zueinander inkompatible Lösungen realisiert haben. Dies stellt besonders für Zulieferer ein immer größer werdendes Problem dar, da diese möglicherweise die gleichen Informationen in vielen verschiedenen Ausprägungen und Formaten bereitstellen müssen. Das Problem wird derzeit durch die Linux Foundation im Rahmen des „Open Compliance Programs“[127] adressiert. Ein Element des Programmes ist die Definition eines Standards zur Auflistung der involvierten Lizenzen und der entsprechenden Urheber. Der Standard trägt den Namen Software Package Data Exchange (SPDX)[128] und definiert ein XML- und ein Excel-basiertes Format. Während das XML-Format für die automatische Verarbeitung durch Werkzeuge gedacht ist, soll das Excel-basierte Format eine Bearbeitung durch Menschen erleichtern. Der Standard verlangt, für jede einzelne Datei eines Projekts den Nachweis ihrer Herkunft und die für sie geltende Lizenz festzuhalten. Die Open-Source-Eingangskontrolle mag sich dem ersten Anschein nach bei den Unternehmen auf die Auswertung solcher Dokumente beschränken - vorausgesetzt, sie wurden von den Lieferanten mit entsprechender Sorgfalt oder unter Zuhilfenahme geeigneter Werkzeuge automatisiert erstellt. Gleichwohl bleibt es letztlich immer die Pflicht des nutzenden Unternehmens, alle Bedingungen einer zu erfüllenden Open-Source-Lizenz auch tatsächlich zu erfüllen. Diese Pflicht auf die Zulieferer zu verschieben, ist letztlich nicht möglich.

7.3.2 Durchführung und Verwaltung von Lizenzinterpretationen

Auf die systematische Erfassung der zur Anwendung kommenden Lizenzen folgt die Interpretation der in den Lizenzen eingeräumten Rechte und auferlegten Pflichten sowie die Verwaltung der Interpretation. Ziel der Interpretation der Lizenzen ist es, Urheberrechtsverletzungen zu vermeiden sowie Klarheit darüber zu bekommen, ob die Lizenzen mit den eigenen Absichten vereinbar sind. Rechtliche Expertise ist bei dieser Tätigkeit unerlässlich. Die Interpretation der Lizenzen ist genauso wie deren Erfassung systematisch durchzuführen und wiederverwendbar zu speichern, um zufällige Fehler auszuschließen. Welche Technologie zur Verwaltung der Interpretationen verwendet wird, ist frei wählbar. Eine Datenbank-orientierte Lösung ist aber sinnvoll, besonders dann, wenn der Prozess der Erfassung und Ausleitung der zu erfüllenden Lizenzbedingungen aus Gründen der Effizienz weitestgehend automatisiert werden soll. Bestehende Lizenzinterpretationen sind zu pflegen und ggf. an sich neu ergebende Aspekte der Rechtsprechung anzupassen.

7.3.3 Möglichkeiten zur Umsetzung von Lizenzinterpretationen

Der Interpretation der Lizenzbedingungen folgt die Definition der Umsetzung der Interpretationen. Die Umsetzung ist in der Regel spezifisch für ein konkretes Lieferszenario: In diesem Schritt müssen die Spezifika der einzelnen Produkte und Lösungen beachtet werden, in denen die erfassten Open-Source-Pakete ganz oder teilweise mit oder ohne Modifikationen enthalten sind. So kann z.B. die Verpflichtung, den Lizenztext der Software beizulegen, im Falle einer vorhandenen GUI (Graphical User Interface) einfach durch einen Button „show license information“ im Hauptmenü umgesetzt werden. Ist kein GUI vorhanden, können die Lizenztexte z.B. auf der Produkt-CD oder auf einer dem Produkt beiliegenden CD als Textdateien bereitgestellt werden. Es ist sinnvoll, bei der konkreten Umsetzung „Best Practises“ zu definieren und sich so weit wie möglich bei der Umsetzung der Lizenzinterpretationen an diesen auszurichten. Das hat die Vorteile, dass der Aufwand durch eine Art Standardisierung minimiert wird, das Risiko, eine Lizenzinterpretation in einem konkreten Produkt falsch umzusetzen, ebenfalls minimiert wird und ein unternehmensweites einheitliches Konzept im Laufe der Zeit als „Brand“ wahrgenommen wird. Als Beispiel zum letzten Punkt: Allen Produkten des Unternehmens A liegt eine CD mit Namen „Open-Source-Software“ bei, auf der alle Angaben und der entsprechende Source Code der gelieferten Open-Source-Software Pakete enthalten sind.

---

[127] Vgl. Linux Foundation: Open Compliance Program (URL: http://www.linuxfoundation.org/programs/legal/compliance)

[128] Vgl. http://spdx.org – der Standard kommt auch in diesem Leitfaden zur Anwendung

Ein effizientes Verfahren, um die Einhaltung der Lizenzbedingungen für konkrete Produkte und Lösungen sicherzustellen, ist die Behandlung der Lizenzbedingungen als Requirements, welche die einzelnen Lieferungen erfüllen müssen. Dazu wird z.B. die Forderung einer Lizenz, den Lizenztext jeder Lieferung der Software beizulegen, als „mandatory requirement“ des Produktes in das verwendete Requirements Engineering Werkzeug eingepflegt. Die konkrete Art der Umsetzung für das jeweilige Produkt wird in diesem Werkzeug dokumentiert und verfolgt. Das Höchstmaß an Effizienz und Vollständigkeit wird dann erreicht, wenn die Lizenzbedingungen (also das Ergebnis der Tätigkeit „Durchführung von Lizenzinterpretationen“) wie weiter oben empfohlen in einer Datenbank zusammen mit den definierten Best Practises hinterlegt sind. Diese werden entsprechend der in der jeweiligen Softwarelieferung enthaltenen Open-Source-Software automatisch aus der Datenbank ausgelesen und in das Requirements Engineering Werkzeug eingepflegt. Durch diese Art der Automatisierung wird die Vollständigkeit der „License compliance requirements“ pro Produkt oder Lösung sichergestellt und auch deren konkrete Umsetzung ist verfolg- und überprüfbar dokumentiert. Dies wiederum vereinfacht die letzte Tätigkeit „Verifikation der Umsetzungsmöglichkeiten“ deutlich.

7.3.4 Verifikation und Erfassung der Umsetzung zur Lieferung

Die Umsetzung der zu erfüllenden Lizenzbedingungen zu verifizieren und zu erfassen ist aus quantitativen Gründen fast nur noch mit Werkzeugunterstützung möglich. Selbst in kleinen DSL-Routern oder DVD-Playern kommen teilweise ganze Linux-Distributionen mit deutlich mehr als 60 Open-Source-Paketen zum Einsatz. Eine Überprüfung, ob alle Lizenzbedingungen eingehalten werden, ist alleine aufgrund dieser schieren Menge ohne Softwareunterstützung nicht mehr möglich. Wie oben bereits beschrieben, sind alle Tätigkeiten des Lizenz-Managements so weit wie möglich zu automatisieren. Ein Hantieren mit Listen und Tabellen, in die Ergebnisse, Interpretationen und Umsetzungen manuell eingetragen und gepflegt werden müssen, wird selbst bei kleinen Lieferungen zwangsläufig zu Fehlern führen.

Lizenzmanagement ist eine den Einsatz von Software flankierende Tätigkeit, egal um welchen Typ Software es sich handelt – sei es Open-Source-Software oder Closed Software. Wichtig ist, dass das Lizenz-Management für jedes Paket und jedes eingesetzte Release der Pakete durchgeführt wird. Denn die Art und Menge der in einem OSS-Paket zur Anwendung kommenden Lizenzen kann in jedem Release unterschiedlich zum Vorgänger sein. Auch auf sich ändernde Lizenzversionen ist zu achten, da damit neue Lizenzpflichten einhergehen können ebenso wie neue Rechte.

8. Chancen und Herausforderungen

Chancen und Herausforderungen von Open-Source-Software zu thematisieren, heißt zu werten. Denn Vor- und Nachteile sind insofern ganz generell subjektiv, als sie immanent Vor- oder Nachteile **für** jemanden sind. Dennoch können auch die Menschen, die dem Phänomen ‚Open Source Software' ob der eigenen Vorteile ganz grundsätzlich gewogen gegenüberstehen, die Lage insgesamt differenziert betrachten. Die Welt ist selten ‚schwarz-weiß'. Und was Chancen bietet, kann an anderer Stelle immer noch mit Risiken einhergehen. In diesem Sinne will das Autoren-Team, das sich gerne und offen als ein Team von ‚Open-Source-Befürwortern' versteht, es dennoch wagen, ein gewiss subjektives, aber eben auch rundes Fazit zu ziehen:

8.1 Herausforderungen

Ein ganz spezifisches Problem beschwert den praktischen Umgang mit OSS auf eine Weise, die mittelbare Konflikte anstrahlt und das Kernproblem verschleiert: Der Begriff „Open-Source-Software" selbst ist nicht geschützt. Zwar ist das Logo der Open Source Initiative geschützt, doch eben nicht der Begriff „Open-Source-Software". Die OSI hat ihn ohne Frage geprägt und definiert. Und trotzdem darf jeder sein Produkt als Open Source Software bezeichnen, selbst wenn die spezielle Lizenz seines Produktes nicht von der OSI zertifiziert worden ist, und auch, wenn sie nicht den 10 OSI Kriterien genügt. Letztlich kann dieser Begriff also beliebig verwendet werden. Und so ist wahrlich nicht überall ‚Open-Source-Software' drin, wo ‚Open-Source-Software' dran steht. Wenn dann echte OSS mit vermeintlicher kombiniert, vertrieben oder empfohlen wird, entstehen Verwirrungen. Damit zu leben, bleibt eine Herausforderung aller OSS Nutzer.

Ebenso bleibt es gerade den OSS Nutzern nicht erspart, sich mit einer besonderen Diskrepanz auseinanderzusetzen: Wenn man mit OSS umgeht, begegnet man bald der Frage, wie denn eine quelloffene Software Sicherheit gewährleisten kann, wenn doch alle Möglichkeiten des Missbrauchs und Ausnutzens von Programmierfehlern dem Konzept nach frei und offen zu Verfügung gestellt werden. Daran gibt es nichts zu deuteln: bei Open Source Software liegen die Dinge – dem Wesen von OSS nach – wirklich offen zu Tage, und zwar offener, als bei Closed Source Software. Auch dieses gilt als Nutzer zu sehen. Daraus aber zu schließen, dass OSS unsicherer sei, als Closed-Source-Software, und dass diese darum vom Konzept her immanent sicherer sei, ist irrig. Denn der Nutzer von Closed-Source-Software weiß schlicht nicht, welche Möglichkeiten des Missbrauchs und Ausnutzens von Programmierfehlern darin enthalten sind. Erst recht weiß er nicht, ob diese Missbrauchsmöglichkeiten - absichtlich oder unabsichtlich – von der CSS Herstellern nicht längst schon an dritte weitergereicht worden sind. Aus der Tatsache, dass man etwas nicht sieht, zu schließen, dass es nicht da ist, ist illusionär. Der CSS-Nutzer ist da auf Zusicherungen der Lieferanten. Er ist auf das Vertrauen begrenzt. Ganz anders der OSS-Nutzer. Gerade weil alle Quellen offen liegen, kann er – wenn es ihm so wichtig ist - die Sicherheit der Software selbst überprüfen. Und wenn es ihm nicht so wichtig ist, kann er – wenigstens bei vielfach genutzter OSS - immer noch darauf vertrauen, dass andere diese Möglichkeit der Kontrolle tatsächlich genutzt haben. So zeigt sich denn auch, dass bei quelloffener Software – getriggert durch die Community - die Verbesserungen und Korrekturen oft viel kürzer getaktet zur Verfügung gestellt werden, als bei (kommerziell getriebener) Closed Source Software. Mithin stellt sich der vermeintlich Nachteil bei näherem Hinsehen sogar als Vorteil heraus.

Eine dritte spezifische Herausforderung für OSS-Nutzer betrifft die Möglichkeit, dass durch die Nutzung von Open-Source-Software eigenes „Core-Know-how" wegen des mit der OSS-Lizenz etablierten Copy-Left-Effekts offengelegt werden muss. Geschäftskritische Alleinstellungsmerkmale können auf diese Weise ungewollt ‚generalisiert' werden. Dass diese Möglichkeit besteht, ist zunächst einmal ein Faktum. Gleichwohl stellt sich auch diese ‚Gefahr' bei näherem Hinsehen als nicht so brennend dar: Meistens ist nämlich das, was das eigene Geschäft im Vergleich mit dem Konkurrenten werthaltig als Alleinstellungsmerkmal begründet, vom Umfang gering – auch wenn es als objektive ‚Kleinigkeit' subjektiv von großer Bedeutung ist. Das bedeutet umgekehrt, dass der größere Teil des

eigenen Geschäftes sehr wohl auch in und mit copyleft-behafteter OSS realisiert werden kann, ohne dem Konkurrenten einen ungewollten Vorteil zu bieten. Man muss eben nur genau differenzieren. Allerdings gilt hier wie sonst auch, dass man den Kuchen nicht gleichzeitig haben und essen kann: Wenn man die Vorteile von OSS nutzen will, muss man das OSS-Spiel mitspielen. Mag diese Regeln nicht, verzichtet man auf die OSS Produkte.

Schließlich gibt es Herausforderungen, die dem OSS-Nutzer manchmal als spezielle ‚Probleme' von Open-Source-Software dargestellt werden, obwohl sie doch in gleichem oder ähnlichem Sinne auch die Closed-Source-Software betreffen. Wir listen einige dieser Aspekte auf:

- **Lizenzrechtliche Verwicklungen**: Softwarepakete sind heute komplex. Sie enthalten Komponenten, die nicht notwendigerweise unter der Hauptlizenz weitergegeben werden. In einer Zeit, wo immer mehr kommerzielle Produkte auch im Verbund mit OSS Komponenten an dritte vertrieben werden, entsteht diese Herausforderung bei OSS und CSS gleichermaßen..
- **IPR-Seiteneffekte**: OSS kann – auch ungewollt - IPRs (Intellectual Property Rights) Dritter verletzen. Programmierer können ganz ohne Absicht erteilte Patente benutzen, ohne Patentlizenzen zu erwerben. Allerdings gilt wiederum, dass dies CSS und OSS gleichermaßen betrifft: Beide Arten der Software können die (Patent)rechte anderer verletzen, sodass der Nutzer der Software der bloßen Nutzung wegen deswegen angegangen wird. Bei kommerzieller CSS glaubt man allerdings, in und mit der Gewährleistung des Lieferanten einen ‚pekuniären' Ansprechpartner für derartige mittelbare Schäden zu haben. Ob die pekuniäre Kraft des CSS-Lieferanten ausreicht, diesen Schaden samt des anzusetzenden juristischen Aufwandes tatsächlich abzufedern, bedarf einer gesonderten Überlegung.
- **Entzug von Nutzungsrechten:** Jede Verletzung der Überlassungspflichten birgt die Gefahr des dauerhaften Entzugs der Nutzungsrechte. Dies kann ebenso bei CSS geschehen, wie bei OSS.
- **Geringer wertige Haftungszusagen**: Open Source Software Lizenzen enthalten immer einen sogenannten Haftungsausschluss. Juristischer Besonderheiten wegen, reduziert sich dieser Ausschluss in Deutschland – sehr grob gesagt - auf die Haftungszusagen, die mit einer Schenkung einhergehen. Die kommerziell vertriebene CSS unterliegt da stärkeren Verpflichtungen. Gleichwohl reduzieren sich auch deren Haftungsmöglichkeiten von der geschäftlichen Kraft her, während umgekehrt kommerzielle OSS Distributoren als eines ihrer Geschäftsmodelle gerade die originäre begrenzte OSS-Haftung aufwerten.
- **Unbekannte Folgekosten**: Nicht abgeschätzte und nicht geplante Kosten – wie upgrade- oder sicherungsbedingte Reintegrationsarbeiten, die durch die Verwendung von OSS entstehen, können den Business Case eines Produktes in Frage stellen. Nur gilt sicher auch für die CSS, dass nicht abgeschätzte und nicht geplante Kosten den Business Case eines Produktes in Frage stellen können. Was aber sicher (für beide) nicht gilt, ist, dass notwendigerweise nicht abschätzbare und nicht planbare Kosten auf den Nutzer zukommen. Hier wie da bedarf es eben einer handwerklich sauberen Analyse der Folgen, bei der Verwendung von CSS ebenso, wie bei der Nutzung von OSS.
- **Know-How-Defizite**: Die einfache Verwendung von OSS-Paketen verführt manchmal dazu, den Aufwand für die Integration in das eigene Produkt zu unterschätzen und für jederzeit reproduzierbar zu halten. Allerdings setzt diese Arbeit Fachwissen voraus. Was bei kommerzieller CSS-Nutzung sozusagen ‚automatisch' im Service mitgeliefert und vergütet wird, muss im Hinblick auf die OSS-Pakete auch organisiert werden. Verzichtet man darauf, kommt das dem Verzicht auf den CSS-Support gleich.
- **Veralteter Software**: Open Source Software kann veralten, sei es in der Entwicklung, sei es in der Dokumentation, sei es in der Distribution. Plötzlich steht man als Nutzer vor „toten", nicht mehr gepflegten Paketen. Allerdings gibt es auch auf CSS-Seite den ähnlichen Fall. Auch Firmen können ‚veralten'. Sie können schließen, ihren Fokus verändern, oder aufgekauft werden. Und ebenso plötzlich steht man auch da vor ‚toter' Software. Auf OSS Seite gibt es aber – der Quelloffenheit wegen – wenigstens die prinzipielle Möglichkeit einer ‚Wiederbelebung'
- **Negative Presse**: Im Falle von OSS-Lizenzverletzungen gibt es durchaus die Möglichkeit, dass die Softwareszene kommunikativ heftig reagiert und so dem persönlichen Ruf schädigt.

Umgekehrt stehen dem CSS-Lieferanten bei Verletzung seiner Vertragsbedingungen ebenfalls heftige Mittel zu Verfügung, nur dass diese wohl in erster Linie juristisch sein dürften.

Diese wenigen Beispiele zeigen, dass es durchaus auch bei OSS Herausforderungen gibt, die zu unterschätzen nicht so clever ist. Sie zeigen allerdings auch, dass dem auf CSS Seite entsprechende Parallel-Herausforderungen gegenüberstehen. In Sachen ‚Herausforderungen' ist OSS nicht ganz so besonders, wie gelegentlich behauptet; in Sachen ‚Chancen' aber sehr wohl:

8.2 Chancen

Open-Source-Software bietet Vorteile, die ihr immanent sind – erst durch die damit verbundenen Eigenschaften zeichnet sie sich überhaupt als Open-Source-Software aus:

- **Lizenzgebührenfreiheit**: Um Open-Source-Software einzusetzen, müssen / dürfen keine Lizenzgebühren gezahlt / gefordert werden.
- **Vier OSS-Grundrechte**: Open-Source-konforme Lizenzen räumen den Nutzern der so lizenzierten Software eine Reihe von weitreichenden Rechten ein.. Die wichtigsten sind die „4 Grundrechte", d.h. die Software kopieren zu dürfen, sie verteilen zu dürfen, sie verändern zu dürfen und sie in veränderter Form weitergeben zu dürfen.

Die daraus resultierende volle Kontrolle über den Source Code führt zu zwei weiteren wichtigen Punkten:

- **Qualität**: Open-Source-Software zeichnet sich durch hohe Qualität aus. Die Freiheiten ermöglichen ein Entwicklungsmodell, zu dem ein Peer-Review-Prozess, die Überprüfbarkeit des Codes, Bug-Reporting und -Fixing sowie häufige Releases gehören.
- **Sicherheit und Integrität**: Der Quellcode kann überprüft werden. So können evtl. Backdoors identifiziert werden, was im Hinblick auf die NSA-Affäre nun von gesteigertem Interesse ist.

Auch bei Betrachtung der Art und Weise der Entwicklung hat Open-Source-Software Vorteile:

- **Transparenz**: Gut organisierte OSS-Projekte haben neben einem, für jeden zugänglichen, Source-Code-Control-System wie GIT oder Subversion unterschiedliche Code-Stränge für stabile Releases und die laufenden Entwicklungen, klare Releases, öffentlich verfügbare Mailinglisten, Bugtrackingsysteme, Wikis usw.. Das heißt, ein gutes Open-Source-Software-Projekt wird in der Öffentlichkeit und nicht hinter verschlossenen Türen entwickelt. Die für jedermann zugänglichen Informationen sind nicht gleichbedeutend damit, dass jeder schreibend auf das Source-Code-Control-System zugreifen kann. In vielen OSS-Projekten haben nur wenige Personen direkte Mitwirkungsrechte. Zwar kann jeder mit Änderungen zum Projekterfolg beitragen. Aber oft werden diese ‚Vorarbeiten' überprüft und nur nach Begutachtung von gesonderten Integratoren in die Projektquellen aufgenommen. So wird gewährleistet, dass nur Code, der vorher durch einen Peer Review gegangen ist, in den Source-Tree des Projektes aufgenommen wird. Dies stellt zum einen eine qualitätssichernde Maßnahme dar, zum anderen ist damit und durch das „sign off"-Verfahren gewährleistet, dass jede Codesequenz einer Person zugeordnet werden kann. Dies ist besonders bei Lizenzierungsfragen und urheberrechtsrelevanten Aktionen wichtig.
- **Erhöhte Innovationsgeschwindigkeit**: Viele Open-Source-Software-Pakete realisieren sehr schnell neue Leistungsmerkmale und stellen an sich eine innovative Lösung dar. Dies mag für manche Softwarehersteller ein Nachteil sein, sofern sie mit ihrer Closed Software in Konkurrenz zu den Open-Source-Software-Paketen stehen. Für Hersteller, die diese innovativen Open-Source-Software-Pakete als Bestandteil eigener Produkte integrieren, wiegt der Vorteil aber doppelt. Denn zum einen sind auf diese Weise innovative Leistungsmerkmale in dem Produkt schnell verfügbar und zum anderen werden durch die Entlastung der eigenen Entwickler und des Projektbudgets Ressourcen frei, die in die Entwicklung weiterer Unique Selling Points (USPs) des Produktes investiert werden können.
- **Weiterverwendbarkeit**: OSS erhöht nicht nur die Innovationsgeschwindigkeit, sondern kann

selbst Inkubator für Innovationen sein. So können bestehende OSS-Lösungen Baustein oder Basis für weitere OSS-Produkte sein. Auch hier können Ressourcen geschont bzw. freigesetzt und so in die eigentliche Innovation gesteckt werden.

- **Schnellerer Release**: Produkte, die OSS enthalten, können schneller auf den Markt gebracht werden. Wie bereits vorgehend beschrieben, entlastet der Einsatz von OSS sowohl die eigenen Entwickler als auch das Projektbudget. Hinzu kommt, dass gut gepflegte OSS-Pakete in der Regel auch gut getestet sind und eine höhere Testabdeckung aufweisen als die eigenentwickelten Alternativen. Somit fällt weniger Testaufwand an, um die notwendige Produktstabilität zu erreichen. Dies kommt vor allen Dingen daher, dass das Einsatzspektrum von OSS prinzipiell breiter ausgelegt ist. Bei eigen entwickelten Lösungen ist das Einsatzspektrum oftmals auf ein Produkt oder in wenigen Ausnahmen auf das Tätigkeitsfeld des Unternehmens beschränkt. Werden diese Eigenschaften geschickt ausgenutzt, können Produkte durch die Verwendung von OSS früher auf den Markt gebracht werden.
- **Unabhängigkeit vom Hersteller**: Nutzer von OSS-Paketen haben in der Regel einen größeren Handlungsspielraum. Fehler (Bugs) können auf den Mailinglisten gemeldet werden – oftmals werden „gut“ gemeldete Fehler schnell von der Entwicklergemeinde behoben. Allerdings hat der Meldende keinen Anspruch auf die Beseitigung der Fehler (siehe Risiken). Fehler und Erweiterungen können auch durch eigene Entwickler beseitigt und programmiert werden; oder externe Software-Dienstleister können damit beauftragt werden. Gleiches gilt auch für die Wartung. Außerdem können beauftragte Serviceerbringer leichter ersetzt werden, da die Basis ihres Tuns, die Software, frei zur Verfügung steht.

Als vorteilhafte Aspekte im Unternehmen ergeben sich:

- **Leichte Integration und Anpassbarkeit**: Dadurch, dass der Source Code der OSS-Pakete verfügbar ist, lassen sich diese leicht in die meist heterogene IT-Landschaft eines Unternehmens integrieren und / oder für den spezifischen Einsatz in einem Produkt anpassen. Oftmals hört man Aussagen wie die, dass OSS-Pakete Rohdiamanten seien, die für den Einsatz in Unternehmen und Produkten nur noch geschliffen werden müssten.
- **Kompetenzausbau der eigenen Mitarbeiter**: Eigene Mitarbeiter können durch die Analyse des Source Codes die eigene Programmierkompetenz erhöhen. Viele prominente OSS-Projekte haben Referenzcharakter in Bezug auf die Lösung von programmiertechnischen Herausforderungen. Demzufolge können die in den Projekten realisierten Lösungen als Vorlagen für die Lösung ähnlicher Fragestellungen herangezogen werden.
- **Firmenübergreifende Möglichkeit zur Standardisierung**: Durch die Offenlegung von Quellcode entstehen oft firmenübergreifende Kooperationen. Jede Firma empfindet, dass das, was sie einbringt, weniger wertvoll ist, als das, was sie von den anderen bekommt. So entsteht über die Zeit Software, die von allen Parteien zur Lösung desselben Problems benutzt wird. Und mit dieser gemeinsamen einheitlichen Entwicklung entsteht ein defacto-Standard.

8.3 Koexistenz, Kooperation und Kollaboration

Bliebe noch ein Aspekt zu erwähnen: Auch wenn es in diesem Leitfaden um Open Source Software geht, steht doch außer Frage, dass es auch eine andere Art der Software gibt, die ebenso erfolgreich genutzt und wirtschaftlich vertrieben wird. Sie ist hier schon als ‚Closed Software‘ bezeichnet worden. Ebenso bekannt wie deren Existenz dürfte die Tatsache sein, dass es noch zum Ausgang des letzten Jahrtausends Irritationen im Miteinander dieser Formen gab. Glücklicherweise hat sich dieses gewandelt. Auch in Open Source Kreisen sieht man die Herausforderung, dass geschäftskritische Alleinstellungsmerkmale nicht in allen Fällen über einen Wechsel des Geschädftsmodells in das Arbeitsmodell von Open Source Software überführt werden können. Gleichwohl betont man in Open Souce Kreisen zurecht, dass dieses in sehr viel mehr Fällen möglich ist, als gemeinhin angenommen wird. Umgekehrt haben Closed-Source-Apologeten längst dazu gewechselt, auch Open Source Software zu nutzen, ja mehr noch: auch auf einer permissiv lizenzierten Open Source Software Basis eigene Software zu entwickeln und zu vertreiben. Das alte Gegeneinander hat sich längst in zu einem

Miteinander gewandelt, zunächst zu einer respektvollen Koexistenz, dann zu einer Kooperation und schließlich sogar zu einer Kollaboration.

Wenn man also ein Fazit ziehen muss, dann bietet sich vielleicht dieser Schluss an: Open-Source-Software erfolgreich einzusetzen, heißt letztlich, das kooperative Open-Source-Spiel der freien Nutzung, der freien Weitergabe und Veränderung mitzuspielen. Wer es mitspielt, macht die Erfahrung, dass der eigene Nutzen die wenigen Kosten weit übersteigt, die sich aus dem Prinzip ‚Paying-by-Doing', wie es aus den OSS Lizenzen erwächst, und aus einem angemessenen Risikomanagement ergeben. Und er wird die Erfahrung machen, dass das, was er der Community gibt, aus seiner persönlichen Sicht immer viel weniger wert ist, als das, was er bekommt. Und das schöne ist: jeder Teilnehmer an diesem Spiel kann diese Rechnung aufmachen und sein persönliches positives Saldo feststellen.

9. Abkürzungsverzeichnis

AGB	Allgemeine Geschäftsbedingungen
ASP	Application Service Providing
BGB	Bürgerliches Gesetzbuch
BHO	Bundeshaushaltsordnung
FAQ	Frequently Asked Questions
Gem HVO NRW	Verordnung über das Haushaltswesen der Gemeinden im Land Nordrhein-Westfalen
GWB	Gesetz gegen Wettbewerbsbeschränkungen
HGrG	Gesetz über die Grundsätze des Haushaltsrechts des Bundes und der Länder
LHO NRW	Landeshaushaltsordnung des Landes Nordrhein-Westfalen
OSD	Open Source Definition der OSI
OSI	Open Source Initiative
PatG	Patentgesetz
SäHO	Haushaltsordnung des Freistaates Sachsen
UrhG	Gesetz über Urheberrecht und verwandte Schutzrechte
UWG	Gesetz gegen den unlauteren Wettbewerb
VOL/A	Vergabe- und Vertragsordnung für Leistungen – Teil A für Vergaben öffentlicher Auftraggeber bei Liefer- und Dienstleistungsaufträgen

10. Literatur- und Quellenverzeichnis

Reincke, Karsten / Sharpe, Greg: Open-Source-License Compendium - How to Achieve Open-Source- License Compliance, Darmstadt, Bonn 2015 (im Internet unter: http://opensource.telekom.net/oslic/releases/oslic.pdf)

Laurent, Phillippe “Open-Source- / Content Licenses before European Courts”, EOLE 2012, http://www.eolevent.eu/en/speeches2012

Van den Brande, Coughlan, Jäger (ed.): The International Free and Open-Source-Software Law Book, 2nd edition

Bernard Lamon: Le droit des licences Open Source, Version 1.1 (August 2009), im Internet unter: http://www.bernardlamon.fr/wp-content/uploads/2009/07/livre-blanc-v3-aout-2009.pdf

MPEP: Manual of Patent Examining Procedure, Handbuch für die Prüfer des US-amerikanischen Patentamts, herunterladbar unter www.uspto.gov

Benkard: Patentgesetz, § 1 PatG, Rdz. 107, 108; 10. Aufl.; Verlag C. H. Beck, München 2006.

Schöttle, Hendrik: Der Patentleft-Effekt der GPLv3, in: Computer und Recht 1/2013, S. 1ff.

Ulrich Wuermeling / Thies Deike, „Open-Source-Software: Eine juristische Risikoanalyse“; in: Computer und Recht 2 / 2003, S.87 ff.

Picot, Henriette, „Die deutsche Rechtsprechung zur GNU General Public License”, in: Open-Source-Jahrbuch 2008, S. 184 ff.

Literaturergänzungen:

Cluster Mechatronik & Automation e.V., Open-Source-Software, Leitfaden zum Einsatz in Unternehmen, 2. erweiterte Ausgabe, 2014 (im Internet unter: http://www.cluster-ma.de/publikationen/leitfaden-oss-2-erweiterte-ausgabe/index.html)

Susanne Strahringer (Hrsg.), Open Source – Konzepte, Risiken, Trends, Praxis der Wirtschaftsinformatik, HMD 283, 2012, dpunkt.verlag

International FOSS law review: http://www.ifosslr.org/ifosslr (Trotz fehlender Neupublikationen seit 2014 bietet diese Site einen reichen Fundus an rechtlichen Informationen zu OSS.)

http://www.ifross.de (Diese Site mag wohl OSS-affin sein, bietet dennoch für den deutschsprachigen Raum wichtige Informationen).

http://www.heise.de/open