MASTERARBEIT

OPEN SOURCE IN KOMMERZIELLEN SOFTWAREPRODUKTEN

Welche Aspekte sind bei der Verwendung von Open Source in kommerziellen Softwareprodukten zu berücksichtigen, um das eigene Geschäftsmodell nicht zu gefährden?

ausgeführt am

CAMPUS 02 GRAZ
FACHHOCHSCHULE DER WIRTSCHAFT

Studiengang
Informationstechnologien und Wirtschaftsinformatik

Von: Elke Laber
Personenkennzeichen: 1410320010

Graz, am 30. März 2017 .....

Unterschrift

EHRENWÖRTLICHE ERKLÄRUNG

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benützt und die benutzten Quellen wörtlich zitiert sowie inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

.....

Unterschrift

DANKSAGUNG

Mit dem Tippen dieser Worte geht nicht nur die Erstellung dieser Masterarbeit zu Ende, sondern auch ein Studium, das mich über viele Jahre hinweg begleitet hat. In dieser Zeit hat sich vieles verändert, es gab Höhen und Tiefen, doch summa summarum kann ich mit fester Überzeugung sagen, dass sich alle Erfahrungen die ich in dieser Zeit machen durfte positiv und bereichernd auf mein Leben ausgewirkt haben.

Ich bedanke mich daher bei allen Lektorinnen und Lektoren, sowie allen Mitstudentinnen und Mitstudenten, die ich in den vergangenen Jahren kennenlernen durfte. Besonderen Dank spreche ich dabei Herrn Dr. Stefan Grünwald aus, der für die Anliegen der Studentinnen und Studenten immer ein offenes Ohr gefunden hat. Bei Herrn DI Michael Neuwersch bedanke ich mich sehr für die unkomplizierte Betreuung meiner Masterarbeit und das zeitnahe, konstruktive Feedback. Ebenso möchte ich mich bei meinem Arbeitgeber bedanken, der allen Mitarbeiterinnen und Mitarbeitern des Unternehmens genügend Flexibilität einräumt, um ein berufsbegleitendes Studium aufnehmen und durchhalten zu können. Bei den Expertinnen und Experten bedanke ich mich recht herzlich für die bereitwillige Teilnahme an den Interviews, die mir die Erarbeitung des Praxisteils in vorliegender Form ermöglicht haben. Abschließend spreche ich großen Dank an all meine Freunde, Bekannten und meine Familie aus, die mich in der Zeit meines Studiums durch seelischen Beistand, Vorbereitung auf Prüfungen, Korrekturlesen dieser Arbeit oder in sonstiger Form unterstützt haben.

Nun freue mich auf eine Zeit, an der Wochenenden wieder Wochenenden sind, und die Vielzahl an geparkten Hobbies und Interessen wieder die Bühne betreten darf.
Ebenso freue ich mich weiterhin auf spannende berufliche Herausforderungen, die ich in den letzten Jahren in zahlreicher Form mit meinen Kolleginnen und Kollegen meistern durfte.

The Dilbert Strip of February 23, 2013 – Scott Adams, Universal Uclick

KURZFASSUNG

Die Einbindung von Open Source Softwareprodukten in kommerzielle, proprietäre Softwareprodukte bietet Unternehmen Chancen und Vorteile, birgt aber ebenso Risiken.

Zweck dieser Arbeit ist herauszufinden, wie ein Unternehmen Open Source Softwareprodukte in sein eigenes kommerzielles, proprietäres Softwareprodukt einbinden kann, ohne dadurch das eigene Geschäftsmodell zu gefährden.

Im Fokus steht dabei die Erarbeitung der rechtlichen Hintergründe. Auf technische Aspekte in Bezug auf die Einbindung von Open Source Softwareprodukten wird in dieser Arbeit nicht näher eingegangen. Ebenso wird auf Unternehmen eingeschränkt, die kommerzielle, proprietäre Softwareprodukte herstellen und vertreiben. Ferner wird aus der Summe der gewerblichen Schutzrechte ausschließlich auf den Umgang mit dem Urheberrechtsgesetz näher Bezug genommen.

Mittels Literaturrecherche konnten proprietäre Softwareprodukte sowie Open Source Softwareprodukte näher beschrieben werden. Darauf aufbauend wird auf die Schutzfähigkeit von Software, Softwarelizenzierung sowie die Positionierung von Unternehmen zu Open Source Software näher erläutert. Ebenso werden Präzedenzfälle behandelt, die die Relevanz der Thematik hervorheben. Daran anschließend wurden Expertinnen und Experten zum IST- und SOLL-Zustand in Bezug auf die Einbindung von Open Source Softwareprodukten innerhalb eines Softwareentwicklungsunternehmens und dessen kommerzielles, proprietäres Softwareprodukt befragt.

In Bezug auf die Beantwortung der Forschungsfrage ergab sich, dass das Verständnis des Wesens der Open Source Softwareprodukte sowie deren rechtliche Grundlage notwendig sind, um diese entsprechend des eigenen Geschäftsmodells einbinden und nutzen zu können.

Im praktischen Teil dieser Arbeit wurden ein Open Source Genehmigungsprozess sowie ein Leitfaden für die Produktentwicklungsabteilung des Unternehmens erstellt. Darauf aufbauend wurde das Ausrollen der besagten Artefakte und weitere Folgeschritte geplant.

In weiterer Folge können Unternehmen, die ebenso Open Source Softwareprodukte in ihre eigenen, kommerziellen, proprietären Produkte einbinden, diese Arbeit als Basis heranziehen, um ihre eigenen Prozesse im Unternehmen zu analysieren und anzupassen.

ABSTRACT

Integration of open-source software into commercial, proprietary software products offers companies both opportunities and risks.

This thesis investigates how a company can integrate open-source software into its own commercial, proprietary software product without jeopardizing its own business model. Companies that develop and distribute commercial, proprietary software products are focused on. Technical aspects concerning the integration of open-source software products are not addressed.

First, the legal background is provided, although, -of all industrial property fights, only copyright las is dealt with in detail.

By means of literature research, proprietary software and open-source software products are described. Based on this, the protection of software, software licensing and the positioning of companies to open-source software products is expounded on, and relevant precedents described. Subsequently, software development experts are asked about the current and target status of integration of open-source software into their commercial, proprietary software products.

It is clear that an understanding of the nature of open-source software and their legal basis are necessary to be able to integrate them into a company's business model.

Therefore, in the practical part of this thesis, an open-source approval process and guide for the product development department is created. The artifact implementation process and further follow-up steps are planned.

Companies that integrate open-source software into their own proprietary products can use this thesis as a starting point to analyze and adapt their own processes within their organization.

INHALTSVERZEICHNIS

1 EINLEITUNG

Die Einbindung von Open Source Softwareprodukten in eigene kommerzielle, proprietäre Softwareprodukte bietet Unternehmen Chancen und Vorteile, birgt aber ebenso Risiken. Das Verständnis des Wesens der Open Source Softwareprodukte sowie deren rechtliche Grundlagen sind notwendig, um diese entsprechend des eigenen Geschäftsmodells einbinden und nutzen zu können.

1.1 Aufgabenstellung und Forschungsfrage

Aufgabenstellung dieser Arbeit ist es, die Forschungsfrage zu beantworten, wie ein Unternehmen Open Source Softwareprodukte in das eigene kommerzielle Softwareprodukt einbinden kann, ohne dabei das eigene Geschäftsmodell zu gefährden. Dabei stehen die Auswirkungen der verwendeten Open Source Lizenzen auf das eigene kommerzielle Softwareprodukt im Vordergrund. Zur Eingrenzung des Themengebietes beschränkt sich diese Arbeit auf kommerzielle, proprietäre Softwareprodukte. Des Weiteren werden die Begriffe Open Source Software und Free Software synonym verwendet. Im Zuge der Erhebung bestehender Open Source Lizenzen sollen diese anhand ihrer rechtlichen Aspekte kategorisiert und Unterschiede aufgezeigt werden. Präzedenzfälle, die die Auswirkung von unrechtmäßiger Einbindung von Open Source Softwareprodukten in kommerzielle, proprietäre Softwareprodukte aufzeigen, sollen ebenfalls in dieser Arbeit dokumentiert und analysiert werden. Dieser erarbeitete Hintergrund bietet die Basis für die Analyse des Umgangs mit Open Source Softwareprodukten innerhalb eines Softwareentwicklungsunternehmens, welches ein eigenes, kommerzielles, proprietäres Softwareprodukt herstellt und vertreibt. Darauf aufbauend soll ein Open Source Genehmigungsprozess erarbeitet werden, welcher innerhalb der Produktentwicklungsabteilung des Unternehmens angewandt werden kann. Des Weiteren soll anhand der erlangten Erkenntnisse ein Leitfaden erstellt werden, der die Mitarbeiterinnen und Mitarbeiter des Unternehmens im Umgang mit Open Source Softwareprodukten unterstützt. Der Genehmigungsprozess und der Leitfaden sollen dazu dienen, mögliche Risiken, die durch Open Source Softwareprodukte in kommerziellen, proprietären Softwareprodukten entstehen können, zu vermeiden.

1.2 Zielsetzung

Im Zuge dieser Masterarbeit sollen folgende Ziele erreicht werden:

1. Rechtliche Rahmenbedingungen von Open Source Lizenzen sind dokumentiert und hinsichtlich der Auswirkung auf deren Einbindung in kommerzielle, proprietäre Softwareprodukte analysiert.
2. Präzedenzfälle, die sich mit der Auswirkung unrechtmäßiger Einbindung von Open Source Softwareprodukten in kommerzielle, proprietäre Softwareprodukte auseinandersetzen, sind erhoben, dokumentiert und analysiert.
3. Bestehende Prozesse hinsichtlich des Einsatzes von Open Source Softwareprodukten in Bezug auf das kommerzielle, proprietäre Softwareprodukt des Unternehmens sind dokumentiert und analysiert.
4. Ein Vorschlag für einen Open Source Genehmigungsprozess ist ausgearbeitet und an die Entscheidungsträgerinnen und Entscheidungsträger kommuniziert.
5. Ein Vorschlag für einen Leitfaden hinsichtlich des richtigen Umgangs mit Open Source Softwareprodukten ist ausgearbeitet und an die Entscheidungsträgerinnen und Entscheidungsträger übermittelt.

Darauf aufbauend sollen im Anschluss dieser Arbeit folgende weitere Ziele erreicht werden können:

1. Der final abgestimmte Open Source Genehmigungsprozess ist ausgerollt und den Mitarbeiterinnen und Mitarbeitern bekannt.
2. Schulungen hinsichtlich des Open Source Genehmigungsprozesses sind vorbereitet und für alle Mitarbeiterinnen und Mitarbeiter verpflichtend zu absolvieren.
3. Die Ergebnisse dieser Masterarbeit fließen in die Verbesserung der technischen Analyse des kommerziellen, proprietären Softwareprodukts ein.
4. Die Ergebnisse dieser Masterarbeit fließen in die Verbesserung der Dokumentation des kommerziellen, proprietären Softwareprodukts ein.
5. Die Ergebnisse dieser Masterarbeit können auf andere kommerzielle, proprietäre Softwareprodukte angewandt werden.

1.3 Methode und Ablauf

Grundlage für den theoretischen Teil dieser Arbeit bildet die Literaturrecherche, durch welche die Hintergründe zu Open Source Software und proprietärer Software erarbeitet werden. Anschließend sollen Open Source Lizenzen durch die vorangegangene Recherche auf die Auswirkungen und Risiken hinsichtlich der Einbindung in kommerzielle, proprietäre Softwareprodukte eingeordnet werden können. Durch die darauffolgende Recherche, Dokumentation und Analyse von Präzedenzfällen, die die Auswirkungen unsachgemäßer Einbindung von Open Source Softwareprodukten in kommerzielle, proprietäre Softwareprodukte aufzeigen, können die bereits erhobenen Auswirkungen und Risiken verdeutlicht werden.

Anschließend findet eine Befragung von Expertinnen und Experten des Unternehmens statt. Diese beschäftigt sich mit dem Thema Open Source Software, genauer mit dem Problembewusstsein zu Open Source Softwareprodukten in kommerziellen, proprietären Softwareprodukten. Durch diese Befragung soll ein IST-Zustand des Umgangs mit Open Source Softwareprodukten innerhalb des Unternehmens erhoben, und ebenso Verbesserungspotentiale aufgedeckt werden können. Diese sollen anschließend zusammen mit dem theoretischen Teil dieser Masterarbeit als Basis für die Erstellung eines Open Source Genehmigungsprozesses dienen. Dieser soll innerhalb der Produktentwicklungsabteilung des Unternehmens angewandt werden. Zusätzlich soll ein Entwurf für einen Leitfaden erstellt werden, welcher die Mitarbeiterinnen und Mitarbeiter des Unternehmens im Umgang mit Open Source Softwareprodukten unterstützt. Diese beiden Artefakte sollen mit Fertigstellung dieser Arbeit an die Rechtsabteilung des Mutterkonzerns des Unternehmens übergeben werden.

1.4 Motivation

Durch meine Tätigkeit als Produktmanager für ein kommerzielles, proprietäres Softwareprodukt in einem wachsenden Entwicklungsumfeld, beschäftige ich mich täglich mit der Realisierung ambitionierter Marktvisionen. Einerseits ist es meine Aufgabe, mich mit der Realisierung neuer Funktionalität zu befassen, andererseits zählt es zu meinen Kompetenzen, über Produktrisiken Bescheid zu wissen, die im schlimmsten Fall das gesamte Geschäftsmodell des Unternehmens gefährden könnten. Der richtige Umgang in Bezug auf die Einbindung von Open Source Softwareprodukten ist hier ein wichtiger Faktor. Durch die Relevanz dieses Themas bin ich bestrebt, mittels dieser Masterarbeit den theoretischen Hintergrund zum Thema Einbindung von Open Source Softwareprodukten in kommerzielle, proprietäre Softwareprodukte aufzuarbeiten, und in der Praxis anzuwenden.

2 SOFTWARE

In folgendem Abschnitt wird der Begriff „Software“, dessen Geschichte und Charakteristiken näher erklärt, um Basiswissen für die Nachvollziehbarkeit der darauf aufbauenden Masterarbeit bereitzustellen.

2.1 Was ist Software?

1958 wurde der Begriff „Software“ zum ersten Mal als Gegenstück zu dem bereits bekannten Begriff „Hardware“ verwendet. Er diente der Bezeichnung aller nicht physischen Elemente eines Computers, somit der Beschreibung aller elektronisch gehaltenen Daten. Diese Definition alleine ist allerdings ungenügend und Bedarf weiterer Spezifikation. (Lichter & Ludewig, 2013)

Im Allgemeinen wird der Begriff „Software“ als Sammelbegriff für (Computer-)Programme und deren dazugehörigen Daten beschrieben (Lassmann, 2006). Des Weiteren können einer Software Beiwerke wie Dokumentation oder Anleitungen hinzugefügt sein. Mittels Software wird das Verhalten der physischen Komponente, der Hardware, wie etwa ein Laptop oder auch eine Waschmaschine, bestimmt, da diese die Software ausführt, genauer die Software abarbeitet (Bruegge, et al., 2004). Damit eine Hardware also bestimmte Tätigkeiten ausführen kann, muss die Software, mit der die Hardware gesteuert wird, alle nötigen Informationen und Daten enthalten, um dies der Hardware zu ermöglichen. (Freund, 2006)

Obwohl Software auf einem Trägermedium hinterlegt werden kann, ist sie von diesem unabhängig anzusehen und daher immateriell, da das Trägermedium dem Namen nach lediglich des Transports der Software dienlich ist, jedoch keinen direkten Einfluss auf das Wesen der Software hat. (Lichter & Ludewig, 2013)

Da Software durch kreatives Arbeiten von einer Entwicklerin bzw. einem Entwickler oder mehreren Entwicklerinnen bzw. Entwicklern erstellt wird und ein immaterielles Produkt darstellt, ist diese durch das Urheberrechtsgesetz geschützt. Der erstellte Text, der bei der Programmierung entsteht, wird als Source Code bezeichnet. Dieser Source Code wird in weiterer Folge durch Kompilierung zu einem Softwareprogramm, welches von einem Computer ausgeführt werden kann. (Schaaf, 2013)

2.2 Historischer Überblick

Bis der Begriff „Software“ 1958 von John W. Turkey eingeführt wurde, verstand man die heutige Form von Hardware und Software als Gesamtpaket. Dabei war die heutige Software als Programmcode der betreffenden Hardware zu verstehen. Endgültige Separation der beiden Begriffe erfolgte durch einen Beschluss der US-Regierung in den 1970er Jahren, welcher festlegte, dass Hardware und Software separat voneinander zu verrechnen seien. Dieses Gesetz legte auch den Grundstein für die Entstehung von Firmen, die sich ausschließlich mit der Erzeugung von Softwareprodukten beschäftigten. (Capers, 2014)

Die Erfolgsgeschichte der Softwareindustrie begann mit der Entwicklung von Rechnern, die mittels Binärcodierung gesteuert werden. Der „Z3“, ein auf elektromagnetischer Grundlage arbeitender Computer, wurde 1941 von Konrad Zuse vorgestellt. Dieser bildete die architektonische Grundlage für spätere Entwicklungskonzepte. 1943 begann man in den USA mit Relaisrechnern zu arbeiten. „ENIAC“ war der erste vollelektronische Rechner, da bei diesem Röhren anstelle von mechanischen Relais eingesetzt wurden. ENIAC wies die etwa 100-fache Leistungsfähigkeit gegenüber dem Z3 auf. (Bruegge, et al., 2004)

John von Neumann arbeitete in den Jahren 1945 bis 1948 an der Princeton Universität an Rechnern mit datengesteuertem Programmablauf sowie deren Programmspeicherung. Von Neumann, welcher für seine Entwicklungskonzepte unter anderem auf der Forschung von Zuse aufbaute, trug daher wesentlich zur Entwicklung des Universalrechners bei. Weitere Meilensteine in Forschung und Entwicklung stellten in den Jahren 1948 bis 1956 der Einsatz von Transistoren, Ferritkernspeichern, Magnetbandspeichern sowie noch heute gebräuchliche Plattenspeicher dar. Ab dem Jahr 1964 war die Halbleitertechnologie durch den Einsatz von Transistoren in integrierten Schaltkreisen sowie durch die Entwicklung von Halbleiterspeichern für weitere Erfolge ausschlaggebend. (Bruegge, et al., 2004)

Die International Business Machine (IBM) Corporation erlangte durch die Nutzung der geleisteten Forschung und Entwicklung dabei schnell die Vorherrschaft am Markt für Großrechner. Obwohl sich bereits 1955 einige IBM Mitarbeiter mit Gründung des ersten Softwareunternehmens für individuelle Softwareanpassungen, der Computer Usage Company (CUC), selbstständig machten, konnte bis zu Beginn der 70er Jahre nur bedingt von einer eigenständigen Software-Industrie gesprochen werden. IBM als Marktführer bot Software und Hardware gebündelt an. Updates, Hilfsprogramme und der Zugriff auf Software Bibliotheken waren dabei für die Kunden von IBM im Kaufpreis inkludiert. (Bruegge, et al., 2004)

Als erstes Unternehmen welches unabhängige Standardsoftware anbot gilt die Applied Data Research (ADR), welches 1964 ein Softwareprodukt zur Erstellung von Flussdiagrammen auf den Markt brachte. Der Großrechner IBM 360, welcher 1965 vorgestellt wurde, setzte sich schnell durch dessen Skalierbarkeit und Flexibilität als Standard durch. Die damit einhergehende große Verbreitung des Modells bot somit erstmals die Möglichkeit für unabhängige Softwareentwicklerinnen und Softwareentwickler, selbstständig größere Softwareprojekte umzusetzen. (Bruegge, et al., 2004)

Ein Kartellverfahren, welches 1968 gegen IBM eingeleitet wurde, hatte die Trennung von Hardware und Software zur Folge. Dies gilt als Grundstein für die Entstehung einer eigenständigen Softwareindustrie. Das Erscheinen von Minicomputern, welche unter anderem durch Unternehmen wie Hewlett-Packard oder Varian angeboten wurden, bot erneute Potentiale für die Softwareindustrie, da auch hier Software unabhängig von Hardware vertrieben wurde. Mit den Minicomputern entstanden auch die ersten Versuche Betriebssysteme zu entwickeln, welche auf unterschiedlicher Hardware ausführbar sein sollten. (Bruegge, et al., 2004)

Den Grundstein für den Software Massenmarkt stellte der 1981 erschienene IBM PC dar. Da es Hardware Herstellern nicht mehr möglich war, alleine ein ausreichend breites Spektrum an Software anzubieten, festigte sich die unabhängige Softwareindustrie. (Bruegge, et al., 2004)

2.3 Softwaregliederung

(Lipinski, 2016) unterteilt den Begriff „Software“ zunächst in Systemsoftware und Anwendungssoftware. Bei ersterem handelt es sich um Software, die der Hardware dienlich ist, wie beispielsweise das Betriebssystem eines Laptops. Die Anwendungssoftware dient dem eigentlichen Anwendungsfall, ist also Mittel um der Nutzerin bzw. dem Nutzer die Lösung einer konkreten Problemstellung zu ermöglichen. Diese gliedert sich weiter in Standardsoftware und Individualsoftware, wobei sich erstere nochmals in anwendungsunabhängige, anwendungsbezogene und branchenabhängige Standardsoftware unterteilt.

Die ISO/IEC-Norm 2382 sieht neben Systemsoftware und Anwendungssoftware noch eine zusätzliche Kategorie „Unterstützungssoftware“ vor. (Herzwurm, 2006)

(Deister & Meyer-Spasche, 2009) beschreiben eine ähnliche Einteilung der Software. Neben den Kategorien der Systemsoftware, welche Betriebssystem und Gerätetreiber beinhaltet, und der Anwendungssoftware wird hier auch die Kategorie der Programmiersprachen gelistet. Programmiersprachen können als Werkzeug verstanden werden, welche die Entwicklung von Software unterstützen bzw. ermöglichen. Daher kann man diese der Kategorie der Unterstützungssoftware nach (Herzwurm, 2006) zuordnen.

Folgende Grafik stellt eine zusammengefasste Sichtweise aller genannten Autoren dar:

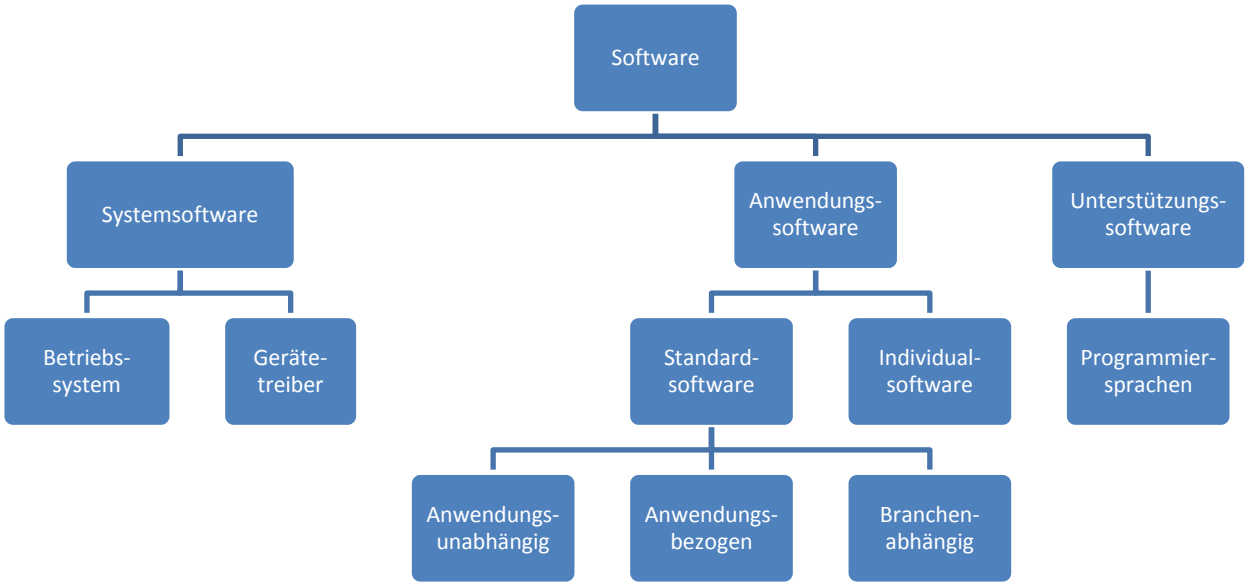

Abbildung 1 – Softwaregliederung

Die genannten Autoren unterteilen Software somit nach deren Art der Verwendung.

(Schaaf, 2013) gliedert Software wie folgt, wobei alle Unterkategorien von Open Source als auch die Unterkategorie „Binär und Source Code“ das Charakteristikum des verfügbaren Source Codes aufweisen. (Jaeger & Metzger, 2016) bezeichnen proprietäre Software, bei der der Source Code zur Verfügung gestellt wird, als „Shared Source Software“.

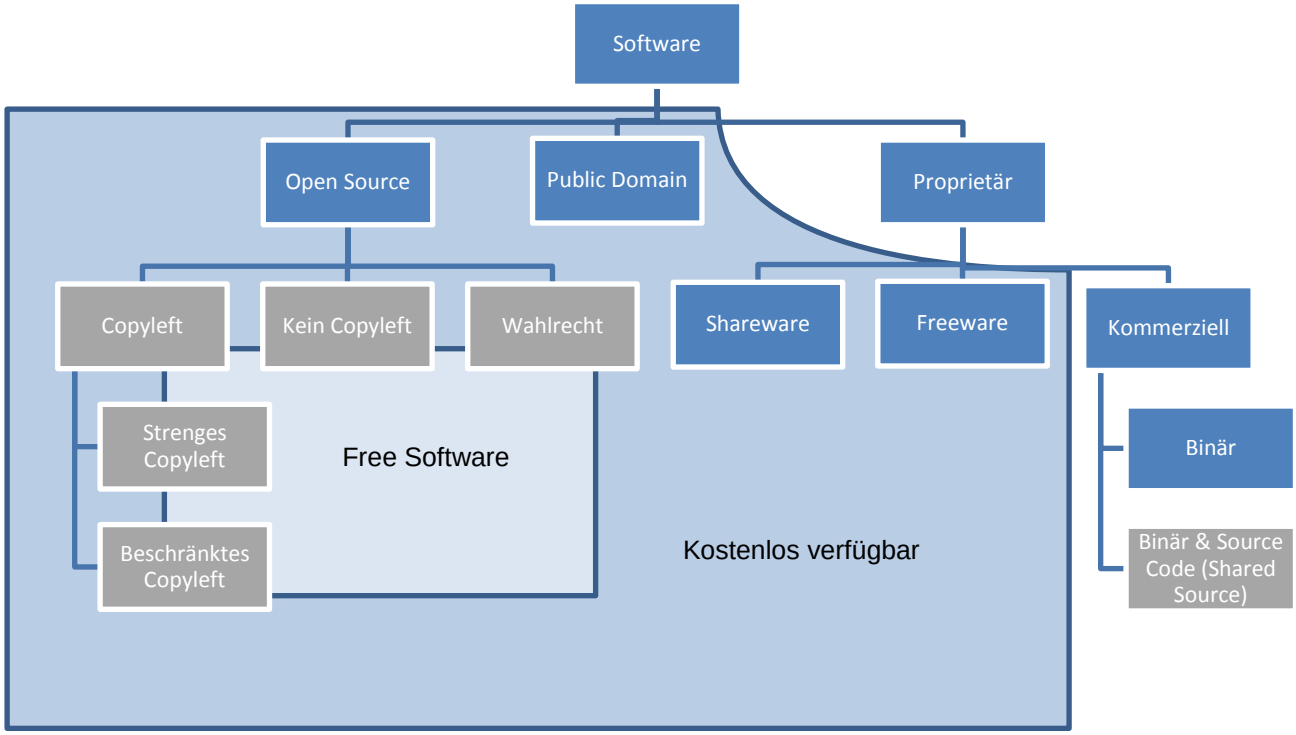

Abbildung 2 - Softwaregliederung nach (Schaaf, 2013)

(Schaaf, 2013) gliedert Software somit nicht nach deren Anwendung, sondern nach deren rechtlichen Rahmenbedingungen.

2.3.1 Proprietäre Software

Proprietäre Softwareprodukte können der Darstellung zufolge kostenlos oder kostenpflichtig vertrieben werden. Der Kern ihres Wesens liegt aber nicht in erster Linie an der Bepreisung des Softwareprodukts, sondern in der alleinigen Entscheidung der Urheberin bzw. des Urhebers, auf welche Art und Weise und in welchem Umfang das Softwareprodukt verwertet werden soll. Da der Source Code für den Hersteller ein schützenswertes Gut darstellt, ist dieser im Normalfall nicht frei zugänglich und auch die Veränderung und Verbreitung sind der Nutzerin bzw. dem Nutzer untersagt. (Schaaf, 2013)

Mit **kommerzieller** Software verfolgt ein Unternehmen das Ziel, mit dem entwickelten Softwareprodukt Umsätze und in weiterer Folge Gewinne zu erzielen. Obwohl der Source Code im Regelfall geheim gehalten wird, gibt es dennoch kommerzielle Softwareprodukte, bei denen der Source Code zugänglich ist. (Schaaf, 2013)

Das **Shared Source** Lizenzmodell wurde 2001 ursprünglich von Microsoft als kommerzielle Antwort auf Open Source Software ins Leben gerufen. Microsoft reagierte dabei auf vermehrte

Kundenanfragen hinsichtlich der Zugänglichkeit zum Source Code. Den Kundinnen und Kunden wird von Microsoft in vielen Fällen eine lizenzgebührenfreie Nutzung von Shared Source Softwareprodukten gestattet. Bei der weiteren Vermarktung von Softwareprodukten, welche Shared Source Software enthalten, ist allerdings eine Gebühr an Microsoft zu entrichten. (Jaeger & Metzger, 2016)

Unter dem Begriff **Freeware** wird zumeist proprietäre Software verstanden, welche kostenlos zur Verfügung gestellt wird. Der Source Code wird hier nicht frei zugänglich gemacht, und der Nutzerin bzw. dem Nutzer ein einfaches Verwertungsrecht eingeräumt. Dieses erlaubt der Nutzerin bzw. dem Nutzer ähnlich des Vermarktungskonzepts von Shareware, freies Kopieren und Verbreiten. Veränderungen dürfen allerdings zumeist nicht durchgeführt werden. Des Weiteren kann es unterschiedliche Bedingungen hinsichtlich privater und betrieblicher Nutzung geben. (Jaeger & Metzger, 2016) (Schaaf, 2013)

Shareware stellt eine besondere Vermarktungsstrategie für kommerzielle Softwareprodukte dar. Die Software soll durch freies Kopieren und Verbreiten möglichst vielen Nutzerinnen und Nutzern zugänglich gemacht werden, ohne dass diese zunächst Lizenzgebühren entrichten müssen. Die Software wird dabei nur mit eingeschränkter Funktionalität oder zeitlich beschränkt zur Verfügung gestellt. Möchte die Nutzerin bzw. der Nutzer die Software in vollem Umfang oder zeitlich unbeschränkt nutzen, hat diese bzw. dieser dafür Lizenzgebühren zu entrichten. (Jaeger & Metzger, 2016) (Schaaf, 2013)

2.3.2 Public Domain Software

Public Domain Software weist keinen oder in Teilen keinen Urheberrechtsschutz auf. Im Gegensatz zur amerikanischen Rechtsprechung kann laut europäischem Recht das Urheberrecht nicht übertragen werden, und es ist auch nicht möglich, vollständig darauf zu verzichten. Daher wird eine Software in der Europäischen Union erst nach Ablauf des urheberrechtlichen Schutzes zu Public Domain Software (Jaeger & Metzger, 2016). Eine Lizenz für Public Domain Software stellt aus europäischer Sicht für die Nutzerin bzw. den Nutzer ein einfaches Nutzungsrecht dar, welches ihr bzw. ihm erlaubt, die Software uneingeschränkt zu nutzen und zu vervielfältigen. Da der Source Code von Public Domain Software nicht frei zugänglich sein muss, kann diese nicht den Open Source Softwareprodukten zugeordnet werden. Auch umgekehrt kann Open Source Software nicht Public Domain Software zugeordnet werden, da der Nutzerin bzw. dem Nutzer erlaubt wird, die Software zu verändern und zu verwerten. Dabei findet das Urheberrechtsgesetz Anwendung, auf welches laut Definition von Public Domain Software zu verzichten ist. (Schaaf, 2013)

2.3.3 Open Source Software

Free Software und Open Source Software werden in der Praxis, und auch in weiterer Folge in dieser Arbeit, als Synonyme gebraucht, obwohl dies bei genauerer Betrachtung nicht ganz exakt ist. (Schaaf, 2013)

Free Software verkörpert den ideologischen Anspruch der Free Software Foundation, welche sich für den freien Umgang mit Software und der Unabhängigkeit von deren Hersteller einsetzt. Der Source Code der Free Software ist frei zugänglich und darf ebenso uneingeschränkt benutzt, verändert und verbreitet werden. (Schaaf, 2013)

Open Source verfolgt die Ansprüche von Free Software, ist in Abgrenzung zu dieser aber mehr als Entwicklungsmodell für gute Software als eine ideologische Haltung zu betrachten. Dabei ist der freie Zugang zum Source Code als Grundvoraussetzung anzusehen. Free Software ist daher eine Teilmenge von Open Source Software. (Schaaf, 2013)

Gerade der Begriff Freeware zeigt jetzt schon deutlich die Zwiespältigkeit der Bezeichnung „free“, also frei, da bei Freeware ausdrücklich frei im Sinne von kostenfrei zu verstehen ist. (Jaeger & Metzger, 2016)

2.4 Softwarelizenzierung

Softwarelizenzierung ist einerseits dem Bereich des Softwarevertriebs zuzurechnen, andererseits soll Softwarelizenzierung verhindern, dass die gegenständliche Software unrechtmäßig genutzt wird. Daher enthält der jeweilige Softwarelizenzvertrag die Rechte und Pflichten, die mit dem Erwerb einer solchen Lizenz verbunden sind. (Schaaf, 2013)

2.4.1 Lizenzmanagement

Softwarelizenzmanagement ist nach (Schaaf, 2013) das Verwalten und prozessuale Organisieren von Softwarelizenzen. Dies ist von vorwiegend kaufmännischem Charakter geprägt, da es die Aufgabe der Softwarelizenzmanagerin bzw. des Softwarelizenzmanagers ist, besondere Kenntnisse über die Softwarelizenzen der Lieferanten zu haben. Für diese sind Prozesse zu schaffen, die den richtigen Umgang mit den entsprechenden Softwarelizenzen im eigenen Unternehmen sicherstellen. Um die Organisationsstruktur und die Softwarelandschaft effektiv verwalten zu können, werden im Bereich des Softwarelizenzmanagements folgende Ziele verfolgt:

- Schaffung von Transparenz
- Planung der Kosten
- Dokumentation der Lizenzen und Prozesse
- Analyse und Bewertung

Ein Unternehmen muss wissen, welche Software es im Einsatz hat. Durch genaues Identifizieren und Prüfen aller Kosten ist es möglich Mehraufwände hinsichtlich Wartung und möglicher Folgekosten zu vermeiden. Kernziel der Dokumentation der Softwarelizenzen, sowie deren Analyse und Bewertung ist es, die Rechtmäßigkeit der verwendeten Softwarelizenzen gegenüber deren Herstellern sicherzustellen und nachweisen zu können. (Schaaf, 2013)

2.4.2 Nutzungsrechte

Fasst man die Erläuterungen der Softwaregliederung aus Kapitel 2.2 zusammen, so ergibt sich nach (Schaaf, 2013) folgende Übersicht:

Nutzungsrecht	Kostenlos	Kopierbar	Modifizierbar	Quelltext offen	Lizenz vorhanden
Proprietär	Selten	Nein	Nein	Nein	Ja
Kommerziell	Nein	Nein	Nein	Nein	Ja
Shareware	Nein	Ja	Nein	Nein	Ja
Public Domain	Ja	Ja	Selten	Selten	Nein
Freeware	Ja	Ja	Nein	Nein	Ja
Free Software	Ja	Ja	Ja	Ja	Ja
Open Source	Ja	Ja	Ja	Ja	Ja

Tabelle 1 - Eigenschaften von Nutzungsrechten

Während im Urheberrechtsgesetz von Nutzungsrechten gesprochen wird, ist der Begriff „Lizenz“ von praktischer Relevanz. Die Urheberin bzw. der Urheber einer Software kann über diese definieren, welche Nutzungsrechte sie bzw. er an ihrer bzw. seiner eigens erbrachten geistigen Leistung an Dritte weitergeben möchte. Gleichermaßen werden über Lizenzen definiert, welche Urheberrechte geschützt werden. Durch das Akzeptieren des Lizenzvertrages stimmt die Nutzerin bzw. der Nutzer den in dem Vertrag genannten Bedingungen zu. Diese geben auch Auskunft über mögliche Vertragsstrafen, mit denen die Nutzerin bzw. der Nutzer bei Verstoß gegen die Vertragsbedingungen konfrontiert sein wird. Das Einwilligen beider Parteien zum Lizenzvertrag ist für dessen Zustandekommen erforderlich, jedoch ist dies in der Praxis mit der Zustimmung der Nutzerin bzw. des Nutzers bei der ersten Installation bereits abgehandelt. (Schaaf, 2013)

Der Kopierschutz ist nicht gleichzusetzen mit dem Begriff der Softwarelizenzierung. Vielmehr dient der Kopierschutz dem Hersteller der Software dazu, unerlaubte Kopien durch die Nutzerin bzw. den Nutzer zu verhindern. Der Kopierschutz stellt also ein unterstützendes Werkzeug dar, um die Einhaltung der Lizenzvertragsbedingungen besser kontrollieren zu können. (Brückner, 2013)

2.4.3 Lizenzvertrag

Der Lizenzvertrag enthält die Nutzungsrechte, welche die Lizenzgeberin bzw. der Lizenzgeber der Lizenznehmerin bzw. dem Lizenznehmer einräumt. Des Weiteren sind Gegenleistungen der Lizenznehmerin bzw. des Lizenznehmers und potentielle Vertragsstrafen bei Verstoß gegen die Vertragsbedingungen, Gegenstand des Lizenzvertrags. Alleinig die Lizenzgeberin bzw. der

Lizenzgeber bestimmt über das Ausmaß des Lizenzvertrages, welcher ebenso für verschiedene Lizenznehmerinnen bzw. Lizenznehmer unterschiedlich ausfallen kann und darf. (Schaaf, 2013)

2.4.4 AGB - Allgemeine Geschäftsbedingungen

Allgemeine Geschäftsbedingungen dienen der Standardisierung von Vertragsabschlüssen, wobei durch die AGB ein beschleunigter Ablauf erreicht werden soll. Dabei sind in den AGB dem Namen nach von einer Vertragspartei allgemeine Geschäftsbedingungen definiert, welche von der anderen Partei akzeptiert werden sollen. In welcher Form die AGB dabei definiert sind oder wo sie aufliegen, spielt für ihre Anwendbarkeit keine Rolle. Lediglich die zustimmende Vertragspartei muss explizit auf diese hingewiesen worden sein. Auch dürfen keine Inhalte definiert werden, welche von der zustimmenden Vertragspartei nicht zu erwarten sind. (Schaaf, 2013)

2.5 Fazit

Der Begriff Software steht für Computerprogramme und deren zugehörige Artefakte. Mittels Software kann Hardware gesteuert werden, sie bestimmt also das Verhalten physischer Komponenten. Das Trägermedium zählt nicht zu den Artefakten der Software, sondern dient lediglich des Transports. Da Software durch kreative, geistige Schöpfung erzeugt wird, ist der daraus resultierende Source Code, der durch Kompilierung zu einem Softwareprogramm wird, urheberrechtlich geschützt. Die rechtliche Trennung von Hardware und Software in den 1970er Jahren sowie die Entwicklung des PCs zum Massenprodukt, führten zur Entstehung der Softwareentwicklungsindustrie, wie wir sie in heutiger Form kennen.

Software kann auf unterschiedliche Weise gegliedert werden. Eine Möglichkeit stellt die Gliederung nach Art der Anwendung dar, wonach sich diese in Anwendungssoftware, Systemsoftware und Unterstützungssoftware unterteilen lässt. Die jeweiligen Kategorien wiederum können in sich genauer gegliedert werden, wie in Abbildung 1 zu sehen ist. Eine weitere, und für diese Arbeit relevante Art der Softwaregliederung, ist die Unterteilung nach deren rechtlichen Rahmenbedingungen, wie in Abbildung 2 ersichtlich ist. Hier stellen die Gruppen Open Source Software, Public Domain Software und Proprietäre Software die erste Ebene der Unterteilung von Software dar.

Das Wesen proprietärer Software liegt in der Entscheidungsfreiheit der Urheberin bzw. des Urhebers, auf welche Art und Weise und in welchem Umfang das Softwareprodukt verwertet wird. Daher ist der Source Code proprietärer Software meist nur in binärer Form zugänglich und Nutzerinnen und Nutzern die Veränderung und Weiterverbreitung des Softwareprodukts untersagt. Meist verfolgen Unternehmen mit proprietärer Software kommerzielle Ziele, daher werden diese Begrifflichkeiten in der Praxis oft in Verbindung gebracht oder sogar als Synonym verwendet. Zu proprietärer Software zählen Shared Source Software, Freeware, sowie Shareware, wobei sich diese Bezeichnungen rein auf das Vertriebsmodell bzw. das Geschäftsmodell hinter der proprietären Software beziehen.

Public Domain Software weist keinen oder in Teilen keinen Urheberrechtsschutz auf, wobei dieser Zustand laut amerikanischem Recht entweder durch Verzicht auf das Urheberrecht, oder laut europäischem Recht nach Ablauf des Urheberrechts entstehen kann. Public Domain Software stellt für die Nutzerin bzw. den Nutzer ein einfaches Nutzungsrecht dar, welches uneingeschränkte Nutzung und Verbreitung bewirkt. Da der Source Code von Public Domain Software nicht frei zugänglich sein muss, kann diese nicht den Open Source Softwareprodukten zugeordnet werden. Auch umgekehrt kann Open Source Software nicht Public Domain Software zugeordnet werden, da der Nutzerin bzw. dem Nutzer erlaubt wird, die Software zu verändern und zu verwerten.

Free Software und Open Source Software können als Synonyme verstanden werden, wobei sich der Begriff Free Software auf den ideologischen Aspekt von freier Software bezieht, und der Begriff Open Source Software eher ein Entwicklungsmodell für gute Software auf Basis der Ansprüche von Free Software beschreibt. Das maßgebende Kriterium stellt jedenfalls der frei zugängliche Source Code der Software dar, welcher ebenso uneingeschränkt genutzt, bearbeitet und verbreitet werden darf.

Problematisch ist jedenfalls die Verwendung des Begriffes „frei“ bzw. „free“, da dieser sowohl für die Vermarktung proprietärer Softwareprodukte, als auch für die Wesensbeschreibung von Open Source Softwareprodukten beansprucht wird.

Softwarelizenzierung ist einerseits dem Bereich des Software Vertriebs zuzurechnen, andererseits soll Softwarelizenzierung verhindern, dass die gegenständliche Software unrechtmäßig genutzt wird. Daher enthält der jeweilige Softwarelizenzvertrag die Rechte und Pflichten, die mit dem Erwerb einer solchen Lizenz verbunden sind.

Das Aufgabengebiet der Softwarelizenzmanagerin bzw. des Softwarelizenzmanagers stellt die Organisation und Verwaltung von Softwarelizenzen dar. Dabei muss diese bzw. dieser sowohl besondere Kenntnisse über die Softwarelizenzen im eigenen Unternehmen, als auch über die Softwarelizenzen der Lieferanten haben, um den richtigen Umgang mit Softwarelizenzen durch Prozesse sicherzustellen. Schließlich muss mittels der geschaffenen Prozesse der rechtlich korrekte Umgang mit den Softwarelizenzen nachgewiesen werden können.

Über eine Lizenz kann von der Urheberin bzw. dem Urheber definiert werden, welche Nutzungsrechte an Dritte weitergegeben werden, welche geschützt werden, und welche Strafen bei Verletzung der Lizenzbedingungen zu erwarten sind. Das Einwilligen beider Parteien zum Lizenzvertrag ist für dessen Zustandekommen erforderlich. Der Kopierschutz stellt ein unterstützendes Werkzeug dar, um die Einhaltung der Lizenzvertragsbedingungen besser kontrollieren zu können.

Allgemeine Geschäftsbedingungen dienen der Standardisierung und Beschleunigung von Vertragsabschlüssen, wobei diese von einer Vertragspartei definiert werden, und von der anderen Partei akzeptiert werden sollen. Diese müssen der zustimmenden Partei zugänglich sein, und dürfen keine unüblichen Bedingungen enthalten.

3 SCHUTZFÄHIGKEIT VON SOFTWARE

Ausschlaggebend für den Schutz von Software ist das Urheberrecht. Darüber hinaus bieten das Patentrecht, das Wettbewerbsrecht sowie das Markenrecht weitere Regelungen um Software zu schützen. (Deister & Meyer-Spasche, 2009)

3.1 Gewerbliche Schutzrechte

Durch gewerbliche Schutzrechte wird es der Schöpferin bzw. dem Schöpfer einer geistigen Leistung ermöglicht, geistiges Eigentum zu schützen. Dies soll einerseits andere auf bestimmte Zeit davon abhalten, die eigene geistige Schöpfung nachzuahmen, andererseits ermöglicht ein gewerbliches Schutzrecht, mit dem geistigen Eigentum zu handeln, wie etwa durch Vergabe von Lizenzen. Gewerbliche Schutzrechte beziehen sich auf mehrere Phasen des Produktlebenszyklus, so gelten diese sowohl für die Phasen

- der Produktion,
- der Einfuhr,
- des Vertriebs,
- des Besitzes, oder
- des Gebrauchs.

Dieser Ausführung zufolge hat man in jeder dieser Phase dafür Sorge zu tragen, dass keine gewerblichen Schutzrechte verletzt werden. (WKO, 2017)

Gewerbliche Schutzrechte lassen sich in eingetragene und nicht eingetragene Schutzrechte unterteilen.

Der Vorteil von nicht eingetragenen Schutzrechten, zu welchen das Urheberrecht und EU-Muster zählen, ist, dass diese automatisch existieren und nicht explizit durch die geistige Schöpferin bzw. den geistigen Schöpfer eingetragen werden müssen. Das bedeutet, dass auch keine Eintragungskosten entstehen. Nachteil an nicht eingetragenen Schutzrechten ist allerdings die schwierige Nachvollziehbarkeit, da es keine offiziellen Listen gibt, in welche das definierte Eigentum eingetragen werden kann. Ein Nachweis, dass beispielsweise ein bestimmter Source Code eines Softwareprogrammes eigenes geistiges Eigentum ist, kann sich als schwierig gestalten. Daher ist bei nicht eingetragenen Schutzrechten eigene, genaue Dokumentation empfehlenswert. (WKO, 2017)

Im Gegensatz zu nicht eingetragenen Schutzrechten, sind eingetragene Schutzrechte dem Namen nach eingetragen. Das bedeutet, dass die eigene geistige Schöpfung explizit in eine allgemein zugängliche Liste aufgenommen wird, und von jedem nachvollzogen werden kann. Der Nachteil dabei ist, dass mit der Eintragung im Regelfall Kosten verbunden sind. Die Schöpferin

bzw. der Schöpfer der geistigen Leistung muss daher einschätzen können, in welchem Verhältnis Kosten und Nutzen der Eintragung stehen. Eine Prüfung bereits eingetragener Schutzrechte ist in jedem Fall empfehlenswert, da mit der potentiellen Neueintragung eventuell bereits bestehende Schutzrechte verletzt werden könnten. (WKO, 2017)

Folgende Übersicht laut (WKO, 2017) gibt einen Überblick der relevantesten Schutzrechte:

Schutzrecht	**Beschreibung**	**Schutzdauer**
Marke	• Schutzfähigkeit bezieht sich auf unterscheidbare Merkmale wie: Namen, Logos, Farben, Formen, Klänge. • Bei hohem Bekanntheitsgrad sind diese auch ohne Eintragung geschützt. • Eingetragen werden können Wortmarken, Bildmarken, Klangmarken, Farbmarken, 3D-Marken und Wort-Bildmarken. • Marke muss nicht neu sein.	10 Jahre, verlängerbar
Herkunftsangaben	• Schutzfähigkeit bezieht sich auf geografische Angaben bzw. Ursprungsbezeichnungen.	Zeitlich unlimitiert
Design	• Schutzfähigkeit bezieht sich auf die Gestalt des Produktes. • Form, Farbe, Glanz, Struktur, Verzierung, … • Das Design muss neu sein und eine Eigenart aufweisen. • Eintragung wird auch Muster bzw Geschmacksmuster genannt. Dies ist in der EU drei Jahre lang auch ohne Eintragung geschützt, sofern die Bekanntheit des Designs nachvollziehbar ist.	In Österreich: In 5-Jahresschritten verlängerbar, maximal 25 Jahre; EU weit: Drei Jahre (ohne Eintragung)
Patent	• Erfindungen in allen Bereichen der Technik • Diese muss neu sein und einen bestimmten Grad der erfinderischen Leistung aufweisen.	Maximal 20Jahre

	• Wird in den meisten Ländern amtlich geprüft.	
Gebrauchsmuster	• Erfindungen in allen Bereichen der Technik • Diese muss neu sein und einen bestimmten Grad der erfinderischen Leistung aufweisen. • Wird in den Ländern nicht geprüft. • Abgrenzung zum Patent: Kosten der Eintragung, Laufzeit und Erfindungshöhe geringer, Programmlogik schützbar, existiert nur in einigen Ländern (darunter Österreich)	Maximal 10 Jahre
Urheberrecht	• Titel und Inhalte von Werken der Musik, Literatur, Filmkunst, Malerei, sowie Computerprogramme, … • Grafiken von Marken oder Entwürfe von Designs können ebenso dazuzählen	Bis 70 Jahre nach dem Tod der (letzten) (Mit-) Urheberin bzw. des letzten (Mit-) Urhebers

Tabelle 2 - Gewerbliche Schutzrechte

Gewerbliche Schutzrechte sind in Österreich in verschiedenen Gesetzen geregelt. Zu den wichtigsten zählen laut (WKO, 2017) folgende:

- das Markenschutzgesetz (MarkenSchG),
- das Musterschutzgesetz (MuSchG),
- das Patentgesetz (PatG),
- das Gebrauchsmustergesetz (GMG), sowie
- das Urheberrechtsgesetz (UrhG).

Die darin enthaltenen Regelungen finden sich auch in den Gesetzestexten anderer Länder wieder, jedoch können sich darin enthaltene Bedingungen auch von den österreichischen Gesetzen unterscheiden. Um überregionale Einigkeit zu erzielen, gibt es diesbezüglich laut (WKO, 2017) sowohl EU-Richtlinien und internationale Übereinkünfte wie

- das Europäische Patent,
- das EU Design, sowie
- die Gemeinschaftsmarke.

Schutzrechte können vielfältig verwendet werden. Für die eigene geistige Leistung können mehrere Schutzrechte eingetragen werden. Ebenso kann sich ein Schutzrecht auf mehrere Produkte auswirken. (WKO, 2017)

Im folgenden Abschnitt wird näher auf das Urheberrecht eingegangen, da dieses zu den nicht eingetragenen gewerblichen Schutzrechten gehört, welches Sondervorschriften für Computerprogramme vorsieht.

3.2 Urheberrecht

Das Urheberrecht im Allgemeinen ist das Recht der kreativen Schöpferin bzw. des kreativen Schöpfers zu entscheiden, was mit ihrer bzw. seiner Leistung geschehen darf. Die Art und Weise der wirtschaftlichen als auch ideellen Verwertung obliegt der Urheberin bzw. dem Urheber, welche bzw. welcher diese Rechte allerdings auch weitergeben darf. (AustroMechana, 2016)

Die 1991 von der EU herausgegebene Richtlinie zur Harmonisierung von Software wurde 1993 von Österreich umgesetzt, und besagt, dass für Software ebenso Schutzrechte laut Urheberrechtsgesetz bestehen, welche mittels einer Novelle unter den „Sondervorschriften für Computerprogramme“ definiert wurden. (Fina, 2006)

3.2.1 Geltungsbereich

Der Begriff „Software“ bzw. „Computerprogramm“ stellt den Geltungsbereich der Richtlinie dar, ist allerdings so generisch definiert, dass darunter Software in jeder Form verstanden werden kann. Darunter fällt jede Form von Technologie, Verwendungszweck, Programmiersprache oder Erscheinungsform. Dies stellt sicher, dass die Richtlinie in Zeiten des technologischen Fortschrittes nicht an Gültigkeit verliert bzw. überholt ist. (Fina, 2006)

Selbst das Material, welches zum Entwurf der Software herangezogen wurde, unterliegt den Bestimmungen des Urheberrechtsgesetzes, wenn es zu deren Entstehung konkret beigetragen hat. Unter Entwicklungsmaterialen können daher etwa Ablaufdiagramme oder ähnliche prozessbeschreibende Dokumente verstanden werden. (Fina, 2006)

Nicht unter die Sonderregelung für Computerprogramme fallen allerdings deren Benutzerdokumentationen, da diese nach den allgemeinen Regeln der Werke der Literatur urheberrechtlich geschützt sind. Nicht im Begriff Software bzw. Computerprogramm inbegriffen sind auch Texte und bildliche Erscheinungen, die während der Ausführung der Software in Erscheinung treten. Diese stellen eigenständige Werke dar, die lediglich in die Software integriert wurden und sind in der Regel durch die allgemeinen Regeln des Urheberrechtsgesetzes schützenswert. (Fina, 2006)

Die grafische Benutzeroberfläche zur Bedienung der Software stellt hingegen einen umstrittenen Punkt hinsichtlich der Zuordenbarkeit zum Begriff Computerprogramm und somit des Gültigkeitsbereiches der Sondervorschriften für Computerprogramme laut Urheberrechtsgesetz dar. So wird hier in der Regel nach dem allgemeinen Erscheinungsbild, welches nach den

allgemeinen Regeln des Urheberrechtsgesetzes geschützt ist, und der Struktur der grafischen Oberfläche unterschieden, welche wiederum dem Begriff Computerprogramm zuordenbar ist. (Fina, 2006)

Des Weiteren ist zu erwähnen, dass sich das Schutzrecht nur auf die Erscheinungsform der Software, wie etwa den Source Code bezieht, nicht jedoch auf ihre Funktionalität. Demzufolge steht es jedem frei, selbstständig Software zu erstellen, welche denselben Zweck erfüllt bzw. dieselbe Idee verfolgt. (Schaaf, 2013)

3.2.2 Der Werkcharakter

Kern und ausschließliches Kriterium für die Schutzwürdigkeit eines Computerprogramms und damit die Anwendbarkeit des Urheberrechtsgesetzes ist, dass es sich dabei um ein „Werk im Sinne des Urheberrechtsgesetzes“ handelt. Dies ist erfüllt, wenn die geschriebene Software laut Richtlinie das Ergebnis der eigenen geistigen Schöpfung der Erstellerin bzw. des Erstellers, und damit der Urheberin bzw. des Urhebers ist. Somit dürfen keine weiteren Kriterien wie etwa ästhetische und qualitative Ansprüche oder Mindestmaße herangezogen werden. Alleinig die eigene geistige Schöpfung ist relevant. (Fina, 2006)

Handelt es sich um alltägliche Computerprogramme, worunter Software verstanden wird, für deren Erzeugung weder Kosten noch geistige Mühen entstanden sind, sind diese nicht als Werk im Sinne des Urheberrechtsgesetzes zu verstehen und somit nicht schützenswert. Der Begriff der „Alltäglichkeit“ bietet hier allerdings Interpretationsspielraum. (Fina, 2006)

3.2.3 Die Urheberschaft

Das Schöpfungsprinzip besagt, dass diejenige Urheberin bzw. derjenige Urheber eines Werkes, und somit auch eines Computerprogramms bzw. einer Software ist, die bzw. der diese geschaffen hat. Da dies nur auf physische Personen zutreffen kann, ist ein originärer Erwerb von Urheberrechten durch etwa eine juristische Person oder durch die Auftraggeberin bzw. den Auftraggeber der hergestellten Software nicht möglich. Eine Urheberin bzw. ein Urheber kann sich selbst jederzeit als solche bzw. als solcher deklarieren. Dies stellt unverzichtbares Recht dar. Somit würde auch eine Verzichtsvereinbarung darauf als nichtig einzustufen sein. (Fina, 2006)

Miturheberschaft ist dann gegeben, wenn mehrere physische Personen gemeinschaftlich Software erschaffen haben, die sich nicht sinnvoll partiell auf die Schöpfung der einzelnen Erzeugerinnen bzw. Erzeuger aufteilen lässt. Das Urheberrecht obliegt dann allen Miturheberinnen bzw. Miturhebern gemeinschaftlich, wobei auch jede einzelne bzw. jeder einzelne alleine gegen Verstöße gegen das Urheberrechtsgesetz vorgehen darf. Durch das gemeinschaftliche Innehaben des Urheberrechts muss auch gemeinschaftlich und einstimmig über die Verwertung der Software entschieden werden, wobei gegen unbegründete Unwilligkeit einer Miturheberin bzw. eines Miturhebers von den anderen Miturheberinnen bzw. Miturhebern geklagt werden kann. (Fina, 2006)

Dementgegen steht der Begriff der Werkverbindung, bei welchem dem Wort entsprechend mehrere für sich alleinstehende Werke einzelner Urheberinnen bzw. Urheber miteinander verbunden wurden. Hinsichtlich der Einigkeit über die Verwertung der Werkverbindung gelten allerdings dieselben Bedingungen wie für die Gemeinschaft der Miturheberinnen bzw. Miturheber. (Fina, 2006)

Wer im Sinne des urheberrechtlichen Schutzes von Computerprogrammen begünstig ist, ergibt sich einerseits aus den Bestimmungen des Urheberrechtsgesetzes, welche aufgrund des Diskriminierungsverbotes auf alle Staatsangehörigen von EU-Staaten anzuwenden sind, und andererseits aus Staatsverträgen mit Nicht-EU-Staaten. (Fina, 2006)

3.2.4 Persönlichkeitsrechtliche Regelungen

Neben dem unverzichtbaren Recht, sich jederzeit als Urheberin bzw. Urheber deklarieren zu dürfen, bestehen für Urheberinnen bzw. Urheber von Computerprogrammen auch alle weiteren persönlichkeitsrechtlichen Regelungen laut Urheberrechtsgesetz. Während genanntes Recht explizit auch für jene Urheberinnen bzw. Urheber gilt, welche das Werk als Dienstnehmerinnen bzw. Dienstnehmer im Rahmen ihrer dienstlichen Obliegenheit geschaffen haben, liegt sowohl das Recht auf Urheberbezeichnung als auch das Recht auf Werkschutz auf Dienstgeberseite, sofern nichts anderes vereinbart wurde. Bei diesen persönlichkeitsrechtlichen Regelungen hat alleine der Dienstgeber das Recht daran, das Werk entsprechend seinen Vorstellungen zu benennen, bzw. über das öffentliche Erscheinungsbild des Computerprogramms zu bestimmen. So dürfen, abgesehen von rechtlich zulässigen Änderungen, ohne Zustimmung der Urheberin bzw. des Urhebers keinerlei Änderungen an Erscheinungsform oder der Art und Weise des Computerprogramms durchgeführt werden. (Fina, 2006)

3.2.5 Verwertungsrechte

Da es für die Verwertung von Computerprogrammen keine Sonderregelungen im Urheberrechtsgesetz gibt, gelten hier die allgemeinen Verwertungsrechte für Werke. Diese Verwertungsrechte werden auch als Ausschließlichkeitsrechte bezeichnet, da diese ausschließlich der Urheberin bzw. dem Urheber zugeschrieben werden. Für Verwertungshandlungen bedarf es daher immer der Zustimmung der Urheberin bzw. des Urhebers. Das **Vervielfältigungsrecht** und das **Verbreitungsrecht** stellen unter den Verwertungsrechten wohl die wirtschaftlich bedeutsamsten Rechte für Software dar. (Fina, 2006)

Gegenstand des **Vervielfältigungsrechts** ist das Erstellen von Kopien einer Software in jeglicher Form. Darunter ist auch ausdrücklich das Herunterladen von Software und deren Installation zu verstehen. Selbst das Ausführen der Software ist Gegenstand des Vervielfältigungsrechts und bedarf der Zustimmung der Urheberin bzw. des Urhebers. (Fina, 2006)

Unter dem **Verbreitungsrecht** ist zu verstehen, dass jede Art und Weise, Software der Öffentlichkeit zur Verfügung zu stellen durch die Urheberin bzw. den Urheber genehmigungspflichtig ist. Dieses Verbreitungsrecht ist allerdings mit der ersten rechtmäßigen

Veräußerung der Softwarekopie erschöpft. Das bedeutet, dass die rechtmäßige Käuferin bzw. der rechtmäßige Käufer der Softwarekopie diese ohne weitere Zustimmung der Urheberin bzw. des Urhebers weiterverbreiten darf. Dies bezieht sich auf den Fall, bei dem die Käuferin bzw. der Käufer durch den Erwerb auch Eigentumsrechte an der Kopie erhalten hat und gilt nicht für andere Formen der Überlassungsverträge, wie etwa der Miete. Räumlich erstreckt sich das Verbreitungsrecht auf den europäischen Wirtschaftsraum. (Fina, 2006)

Möchte die Urheberin bzw. der Urheber anderen Personen einzelne oder alle Verwertungsrechte übertragen, so kann sie bzw. er dies mittels einer **Werknutzungsbewilligung** vornehmen. Diese Art und Weise der Übertragung der Verwertungsrechte, sowie die zeitliche und örtliche Reichweite werden von der Urheberin bzw. vom Urheber bestimmt, und können sogar soweit führen, dass die Verwertungsrechte ausschließlich an die Werknutzungsberechtigte bzw. den Werknutzungsberechtigten übertragen werden, sodass selbst die Urheberin bzw. der Urheber sich gleich einer außenstehenden Person hinsichtlich der Nutzung der Software zu verhalten hat. Des Weiteren kann die Übertragung der erhaltenen Werknutzungsrechte laut Urheberrechtsgesetz ohne Zustimmung der Urheberin bzw. des Urhebers erfolgen, sofern vertraglich nichts anderes festgelegt wurde. Um einer Auftraggeberin bzw. einem Auftraggeber die wirtschaftliche Verwertung der in Auftrag gegebenen Software zu gewährleisten, hat diese bzw. dieser laut Urheberrechtsgesetz an der in Auftrag gegebenen Software immer ein unbeschränktes Werknutzungsrecht, sofern nichts anderes vereinbart wurde. (Fina, 2006)

Laut Urheberrechtsgesetz ist die relevanteste Art der **freien Werknutzung**, die genehmigungsfreie Vervielfältigung zum privaten und eigenen Gebrauch, für Software ausgeschlossen. Dies hat zur Folge, dass das Kopieren von Computerprogrammen ohne Zustimmung der Urheberin bzw. des Urhebers nicht erlaubt ist. Das Vervielfältigen durch die berechtigte Nutzerin bzw. den berechtigten Nutzer ist allerdings genehmigungsfrei in dem Ausmaß erlaubt, welches objektiv für die zweckmäßige Nutzung der erworbenen Software notwendig ist. Ansonsten dürfte die rechtmäßige Erwerberin bzw. der rechtmäßige Erwerber das Computerprogramm aufgrund des Erschöpfungsprinzips zwar weiterverkaufen, aber selbst nicht nutzen. Das Ausmaß der zweckmäßigen Benutzung kann allerdings auch vertraglich festgelegt werden. (Fina, 2006)

Des Weiteren erlaubt ist das Erstellen einer **Sicherungskopie** durch die berechtigte Nutzerin bzw. den berechtigten Nutzer, jedoch mit der Einschränkung, dass diese zur zweckmäßigen Nutzung des Computerprogramms notwendig ist. Diese Formulierung ist insofern problematisch, da die Sicherungskopie für die ordnungsgemäße Benutzung der Software nie notwendig ist, sondern lediglich für die Vermeidung bzw. Einschränkung von Folgeschäden bei Beschädigung der zur Nutzung berechtigen Software dienen soll. Eine Notwendigkeit einer Sicherungskopie kann demzufolge nur dann gegeben sein, wenn Software in ihrer erhaltenen physischen Form Gefährdungspotential hinsichtlich Beschädigung aufweist und nicht anders ersetzt werden kann. (Fina, 2006)

3.2.6 Schutzdauer

Software ist für die Urheberin bzw. den Urheber befristet geschützt, und zwar auf die Zeit von 70 Jahren nach dem Tod der Urheberin bzw. des Urhebers. Bei einer Miturhebergemeinschaft startet die Berechnung des Zeitraums ab dem Tod der letzten Miturheberin bzw. des letzten Miturhebers. Dies ist aufgrund der Kurzlebigkeit von Software allerdings nur bedingt von Relevanz. (Fina, 2006)

Mit Ablauf der Schutzdauer ist die Software nicht mehr urheberrechtlich geschützt und stellt Allgemeingut dar. Ab diesem Zeitpunkt gilt sie als Public Domain Software, welche in Kapitel 2.2.1 näher beschrieben ist. (Schaaf, 2013)

3.2.7 Copyright

Das Copyright stellt das amerikanische Pendant zum europäischen Urheberrecht dar. Während aus amerikanischer Sicht jegliche Rechte an Dritte weitergegeben werden können, ist die Übertragung der Urheberschaft aus europäischer Sicht nicht möglich. Dem Copyright zufolge ist es daher möglich, dass sowohl juristische Personen als auch natürliche Personen als Urheberin bzw. Urheber einer nicht selbst erbrachten geistigen Schöpfung aufscheinen, während aus europäischer Sicht nur die tatsächlichen geistigen Schöpferinnen bzw. Schöpfer in Form einer natürlichen Person als Urheberin bzw. Urheber auftreten. Ungeachtet dieses Unterschieds spielt die Übertragbarkeit der Nutzungsrechte die wesentliche Rolle, welche zumeist über Lizenzen geregelt ist. (Schaaf, 2013)

3.2.8 Copyleft

Das Copyleft stellt eine Vertragsklausel in einer Lizenz dar, welche in unterschiedlich starker Ausprägung auftreten kann. Die Klausel stellt im Allgemeinen sicher, dass die Software über ihren ganzen Lebenszyklus hinweg frei verfügbar gemacht, verändert und vervielfältigt werden darf. Des Weiteren stellt die Klausel sicher, dass der Source Code frei zugänglich ist. (Schaaf, 2013) Auf die unterschiedlichen Ausprägungen wird in Kapitel 6.4 näher eingegangen.

3.3 Fazit

Gewerbliche Schutzrechte ermöglichen es, das Eigentum an der eigenen geistigen Schöpfung zu schützen. Dies dient dazu andere von der Nachahmung abzuhalten, und ermöglicht ebenso den Handel mit dem geistigen Eigentum.

Gewerbliche Schutzrechte gliedern sich in eingetragene und nicht eingetragene Schutzrechte. Eingetragene Schutzrechte werden kostenpflichtig in eine allgemein zugängliche Liste aufgenommen, wodurch eine Prüfung von vorhandenen Eintragungen oder von möglichen Verletzungen eingetragener Schutzrechte mittels dieser Liste möglich ist. Dazu zählen in

Österreich unter anderem das Schutzrecht der Marke, der Herkunftsangaben, des Designs, des Patents, des Gebrauchsmusters.

Nicht eingetragene Schutzrechte stellen unter anderem das Urheberrecht sowie das EU Muster dar. Diese gewerblichen Schutzrechte existieren automatisch und müssen daher nicht erst in eine Liste aufgenommen werden. Dadurch fallen auch keine Kosten an. Kritischer Faktor ist hierbei die Nachvollziehbarkeit, da es keine allgemein zugängliche Liste gibt, die überprüft werden kann. Besonders bei Softwareprogrammen ist daher besonderes Augenmerk auf eigenständige Dokumentation zu legen, da man im Streitfall das geistige Eigentum nachweisen können muss.

Gewerbliche Schutzrechte sind in ähnlicher Form in allen EU Ländern vorhanden, dennoch können sich diese in ihren Ausprägungen unterscheiden. Um dieses Problem nach und nach zu lösen gibt es mittlerweile internationale Bestrebungen diese anzugleichen. Beispielsweise gibt es auf EU Ebene das Europäische Patent und die EU-Marke.

Die Schöpferin bzw. der Schöpfer einer geistigen Leistung hat jedenfalls dafür Sorge zu tragen, dass gewerbliche Schutzrechte anderer nicht verletzt werden.

Auf das Urheberrechtsgesetz wird in weiterer Folge näher eingegangen, da dieses zu den nicht eingetragenen Schutzrechten zählt und Sondervorschriften für Computerprogramme enthält.

Urheberschaft bedeutet Entscheidungsfreiheit über die Art und Weise der Verwertung der eigens erbrachten Leistung. Verwertungsrechte können auch weitergegeben werden, allerdings nicht die Urheberschaft. Die Schutzrechte für Software sind unter den „Sondervorschriften für Computerprogramme“ definiert.

Zu Computerprogrammen im Sinne des Urheberrechtsgesetzes zählt Software in jeder Form, sowie dessen Entwurfsmaterial, wenn es zur Entstehung konkret beigetragen hat. Benutzerdokumentation, sowie in die Software integrierte Texte und bildliche Erscheinungen sind den Werken der Literatur zuzurechnen und daher durch die allgemeinen Regelungen des Urheberrechtsgesetzes geschützt. Strittig ist die Zurechnung der grafischen Benutzeroberfläche, dessen Erscheinungsbild zumeist den allgemeinen Regelungen des Urheberrechts unterliegt, die Struktur der grafischen Oberfläche allerdings dem Computerprogramm zuordenbar ist.

Das urheberrechtliche Schutzrecht laut den Sondervorschriften für Computerprogramme bezieht sich alleinig auf die Erscheinungsform, wie etwa den Source Code der Software, und nicht auf deren Funktionalität. Daher ist es legitim selbstständig Softwareprodukte herzustellen, welche dieselbe Funktionalität bieten wie bereits bestehende Softwareprodukte.

Für die Schutzwürdigkeit von Software laut Urheberrechtsgesetz, muss der Werkcharakter vorliegen, was durch die eigene kreative Schöpfung erfüllt ist. Andere Kriterien wie Ästhetik oder Qualität spielen keine Rolle. Lediglich alltägliche Computerprogramme sind von den Schutzrechten laut Urheberrecht ausgenommen, da hierzu Softwareprogramme zählen, für dessen Entstehung keinerlei geistige Mühen entstanden sind. Dies bietet in der Praxis allerdings Interpretationsspielraum.

Urheberschaft stellt ein unverzichtbares Recht einer natürlichen Person dar, daher kann diese weder von einer juristischen Person beansprucht, noch an eine andere natürliche oder juristische Person übertragen werden.

Stellen mehrere Personen gemeinschaftlich ein Softwareprodukt her, liegt Miturheberschaft vor. Jeder Miturheber bzw. jede Miturheberin kann einzeln gegen Urheberrechtsverletzungen vorgehen, über die Verwertung des Softwareprodukts muss allerdings gemeinschaftlich und einheitlich entschieden werden, wobei gegen unbegründete Unwilligkeit der Zustimmung geklagt werden kann. Bei einer Werkverbindung handelt es sich um den Verbund mehrerer Werke verschiedener Urheber, wobei bei der Verwertung der Werkverbindung dieselben Regelungen gelten wie bei Miturheberschaft.

Die Begünstigung im Sinne des urheberrechtlichen Schutzes von Computerprogrammen wirkt sich einerseits auf alle Staatsangehörige von EU-Staaten aus, andererseits ergibt sich diese durch das Abschließen von Staatsverträgen.

Das Recht auf Urheberbezeichnung sowie das Recht auf Werkschutz obliegt dem Dienstgeber, wenn das Werk im Rahmen der dienstlichen Obliegenheit erstellt wurde. Dieser kann das Werk daher nach seinen Vorstellungen benennen und über das öffentliche Erscheinungsbild entscheiden.

Für die Verwertung von Computerprogrammen gelten die allgemeinen Verwertungsrechte für Werke laut Urheberrechtsgesetz. Diese werden auch als Ausschließlichkeitsrechte bezeichnet, da diese ausschließlich der Urheberin bzw. dem Urheber obliegen. Das Vervielfältigungsrecht und das Verbreitungsrecht stellen bei Softwareprogrammen die wirtschaftlich bedeutsamsten Rechte dar.

Mittels einer Werknutzungsbewilligung können Verwertungsrechte eingeschränkt oder auch ausschließlich auf andere Personen übertragen werden. Wurde das Softwareprodukt im Rahmen der dienstlichen Obliegenheit erzeugt, steht der Auftraggeberin bzw. dem Auftraggeber immer ein unbeschränktes Werknutzungsrecht zu.

Die genehmigungsfreie Vervielfältigung zum privaten und eigenen Gebrauch ist laut Urheberrechtsgesetz für Software ausgeschlossen. Ein Kopieren von Computerprogrammen ohne Zustimmung der Urheberin bzw. des Urhebers ist daher untersagt. Das Vervielfältigen im Rahmen der zweckmäßigen Nutzung ist aber zulässig. Ist für die zweckmäßige Nutzung auch das Erstellen einer Sicherungskopie nötig, darf dies vorgenommen werden.

Der urheberrechtliche Schutz von Software erlischt 70 Jahre nach dem Tod der Urheberin bzw. des Urhebers, oder der letzten Miturheberin bzw. des letzten Miturhebers. Danach stellt die Software Allgemeingut dar.

Das Copyright stellt das amerikanische Gegenstück zum europäischen Urheberrecht dar, wobei bei diesem jegliche Rechte übertragbar sind. Das Copyleft ist eine Vertragsklausel, die Auskunft über die Lizenzänderungsmöglichkeit bei Bearbeitung und weiterer Distribution eines Open Source Softwareprodukts gibt. Diese kann in unterschiedlicher Ausprägung auftreten.

4 PROPRIETÄRE SOFTWARE

Das Wesen proprietärer Softwareprodukte, sowie deren Historie und rechtliche Aspekte werden in folgendem Kapitel näher beleuchtet.

4.1 Was ist proprietäre Software?

Ein proprietäres Softwareprodukt ist Eigentum einer natürlichen oder juristischen Person. Um das proprietäre Softwareprodukt zu nutzen, erwirbt man von dieser Person eine Lizenz, die der Nutzerin bzw. dem Nutzer bestimmte Rechte und Pflichten aber auch Verbote einräumt. Im Gegensatz zu Open Source Softwareprodukten erhält man durch den Erwerb von proprietären Softwareprodukten im Normalfall allerdings nicht den Source Code des Softwareprodukts, sondern nur den kompilierten Programmcode. Dies ist der Fall, da die Urheberin bzw. der Urheber im Source Code des Softwareprodukts ein schützenswertes Gut sieht, und sie bzw. er dieses Geschäftsgeheimnis für sich wahren möchte. Alleinig der Urheberin bzw. dem Urheber bleibt es somit überlassen, auf welche Art und Weise die Software vermarktet wird, und wer sie in welcher Form nutzen darf. (Hennig, 2009)

(Weber, 2005) beschreibt das Wesen proprietärer Produkte plakativ mit dem Coca-Cola Beispiel. Jeder kann dieses Getränk kaufen, und kann mittels des Etiketts sehen, welche Grundzutaten sich darin befinden. Die Formel der Zusammensetzung wird allerdings proprietär gehalten, und ist daher nicht zugänglich. Dadurch kann niemand anders die Grundformel von Coca-Cola als Basis heranziehen, um diese weiterzuentwickeln und das Ergebnis weiterzuverbreiten. Gleichermaßen lässt sich dieses Beispiel laut (Weber, 2005) auf Microsoft Windows übertragen. Keine Softwareentwicklerin und kein Softwareentwickler ist in der Lage, dieses Softwareprodukt anzupassen oder weiterzuentwickeln, da der Source Code unter Verschluss gehalten wird.

Mit der Erzeugung kommerzieller Softwareprodukte wird das Ziel verfolgt, Umsätze und daraus entstehende Gewinne durch den Verkauf von Lizenzen zu generieren. Bei dem Großteil der kommerziellen Softwareprodukte handelt es sich um proprietäre Softwareprodukte, es gibt aber auch kommerzielle Softwareprodukte, bei denen der Source Code frei verfügbar ist. Außerdem kann proprietäre Software sowohl kommerziell als auch kostenlos verbreitet werden, daher ist proprietäre Software mit kommerzieller Software nicht gleichzusetzen. (Schaaf, 2013)

4.2 Historischer Überblick

Computer waren bis in die späten 1960er Jahre große Maschinen, die in gekühlten Räumen errichtet wurden. Meist wurden diese schweren Gerätschaften eher vermietet als verkauft. Serviceleistungen und der zugehörige Programmcode waren im Preis des Verkaufs oder der Vermietung enthalten und wurden nicht extra verrechnet. Unter Anbieterinnen bzw. Anbietern als auch Nutzerinnen bzw. Nutzern war es üblich, den Source Code frei zur Verfügung zu stellen, was auch an der damals fragmentierten und eingeschränkten Marktsituation lag. Mit der

Aufteilung in Hardware und Software als eigenständige Güter und dem Auftauchen von standardisierten Mikroprozessoren veränderte sich dieser Markt Ende der 1970er Jahre. Da Software nun unabhängig von Hardware entwickelt und vertrieben werden konnte, wurde das Geheimhalten des Source Codes, und damit der Vertrieb eines proprietären Softwareprodukts, üblich. (Landley, 2009)

Durch die Gesetzesänderung des amerikanischen Copyrights, welches nun auch binärem Code Urheberschutz zusprach, setzte sich proprietäre Software am Markt durch. Microsoft stellte sich als Vorreiter für das kommerzielle Herstellungs- und Vertriebssystem für kommerzielle, proprietäre Softwareprodukte heraus, und auch IBM begann, für eine anwachsende Liste ihrer Softwareproduktpalette den Source Code unter Verschluss zu halten. (Moore, 2001)

4.3 Lizenzmodelle

Eine Lizenz im Allgemeinen ist die Einräumung von Nutzungsrechten (usedSoft, 2016). Lizenzmodelle für proprietäre Softwareprodukte gibt es in großer Zahl. (Krcmar, 2005) teilt die Modelltypen in folgende Kategorien auf:

Modelltyp	**Vorrangige Bezugsgröße**	**Beispiel**
Nutzerbezogene Modelle	Nutzeranzahl	Lizenzkosten pro Bürger
Wertbezogene Modelle	Personalbestand Herstellungskosten	Lizenzen für Personaladministrationssoftware
Zeitbezogene Modelle	Nutzungsdauer	Lizenz pro Kalenderjahr
Infrastrukturbezogene Modelle	Ausmaß der Nutzung der genutzten Infrastruktur	Lizenz pro Device

Tabelle 3 - Lizenzmodelltypen

Da sich der Lizenzwert bei **nutzerbezogenen Modellen** von der definierten Nutzeranzahl ableitet, ist die Umsetzung dieses Modells aufgrund des leicht realisierbaren indirekten Zugriffs über das Internet schwierig. (Hennig, 2009)

Bei einem **wertbezogenen Modell** kann es sich beispielsweise um eine Lizenz für eine Personaladministrationssoftware handeln, bei dem der Lizenzwert pro Personalstammsatz ermittelt wird. (Hennig, 2009)

Zeitbezogene Modelle unterteilen sich grundsätzlich in befristete und unbefristete Modelle. Handelt es sich um eine befristete Lizenz, sind darin üblicherweise auch Vereinbarungen zu Wartung oder Upgrade enthalten. Bei unbefristeten Lizenzen hingegen müssen Dienstleistungen dieser Art über die Lizenz hinaus geregelt werden. (Hennig, 2009)

Dient beispielsweise die Anzahl der sich im Einsatz befindlichen Hardware als Berechnungsgrundlage des Lizenzwerts, so handelt es sich um ein **infrastrukturbezogenes Modell**. (Hennig, 2009)

Auch Mischformen aus beschriebenen Modelltypen sind möglich. (Hennig, 2009)

Ein Kopierschutz der Software ist nicht gleichbedeutend mit einer Softwarelizenz, doch ist dieser gerade bei proprietären Softwareprodukten in den Lizenzierungsmaßnahmen enthalten. (Schaaf, 2013)

Ein End User License Agreement (EULA) findet Anwendung bei proprietären Softwareprodukten und beinhaltet im Regelfall Hinweise und Rahmenbedingungen zur richtigen Verwendung des Softwareprodukts. Auch die Bestimmungen des Urheberrechts werden in der EULA wiederholt. Diese sollen von der Nutzerin bzw. vom Nutzer akzeptiert und eingehalten werden. Das Ziel des Herstellers ist dabei, dass die Nutzerin bzw. der Nutzer nicht nur mit dem Händler des Softwareprodukts einen Vertrag eingeht, sondern über die EULA auch mit dem Hersteller. Dies wird erreicht, indem die Nutzerin bzw. der Nutzer bei der Installation der Software den Vertragsbedingungen der EULA zustimmen muss, ansonsten wird die Installation nicht weiter ausgeführt und die Nutzerin bzw. der Nutzer kann das erworbene Softwareprodukt nicht verwenden. Aus juristischer Sicht ist die EULA an sich umstritten, da laut Urheberrechtsgesetz eine bestimmungsgemäße Nutzung der Software ohne Zustimmung des Herstellers erlaubt ist. (Groll, 2015) (Schaaf, 2013)

4.4 Vorteile und Nachteile proprietärer Software

(Schaaf, 2013) und (Schraml, 2014) listen folgende Vor- und Nachteile proprietärer Software:

Vorteile für Kunden	**Nachteile für Kunden**
Support	Hohe Lizenzkosten
Weiterentwicklung	Abhängigkeit
Marktdurchdringung	Limitierung von Ressourcen
Wahrung des Firmengeheimnisses	Eingeschränkte Zielgruppe

Tabelle 4 - Vor- und Nachteile proprietärer Software

Genauer werden diese wie folgt erläutert.

4.4.1 Vorteile proprietärer Software

Support

Durch die Nutzung eines proprietären Softwareprodukts können von dessen Hersteller Supportleistungen wie die Behebung von Fehlern mittels Updates, Bereitstellung von Schulungen sowie sonstige Unterstützungsmöglichkeiten erwartet werden. Die damit verbundenen Kosten hängen von der vertraglichen Gestaltung ab, würden aber auch bei freien Softwareprodukten anfallen. Für den Hersteller positiv ist, dass benötigter Kundensupport nur von ihm selbst oder von Vertragspartnern durchgeführt werden kann. (Schaaf, 2013) (Schraml, 2014)

Weiterentwicklung

Proprietäre Softwareprodukte werden meist von einem Hersteller erzeugt und kommerziell vertrieben. Dies wiederum bedeutet, dass erwartet werden kann, dass der Hersteller das betreffende Softwareprodukt weiterentwickelt und diese Weiterentwicklungen mittels eines Upgrades genutzt werden können. Abhängig von der Kundenbeziehung besteht auch die Möglichkeit sich mit dem Hersteller über künftige Entwicklungen abzustimmen oder die Entwicklungspläne einzusehen. (Schaaf, 2013) (Schraml, 2014)

Marktdurchdringung

Hat ein Softwareprodukt eine gewisse Marktdurchdringung erreicht und wird von vielen Kunden des Marktes genutzt, so ist davon auch eine gewisse Beständigkeit des Herstellers und des Produktes abzuleiten. Für den Hersteller wiederum stellt ein am Markt stark verbreitetes Softwareprodukt neben den erzielten Einnahmen auch den zusätzlich positiven Effekt der Kundenbindung dar. (Schaaf, 2013) (Schraml, 2014)

Wahrung des Firmengeheimnisses

Wie bereits (Hennig, 2009) und (Weber, 2005) angeführt stellt der Source Code eines proprietären Softwareprodukts für den Hersteller ein schützenswertes Gut dar, wodurch dieser nicht frei zugänglich, sondern nur in binärer Form dem Markt zugänglich gemacht wird. Durch die Wahrung dieses Firmengeheimnisses wird es der Konkurrenz erschwert, ein vergleichbares Softwareprodukt auf den Markt zu bringen.

4.4.2 Nachteile proprietärer Software

Kosten

Mit proprietären Softwareprodukten sind oft hohe Lizenzkosten verbunden. Auch die Kosten der Installation von Upgrades sowie der Bezug von Supportleistungen können durch eine mögliche Monopolstellung des Herstellers höher ausfallen, als in anderen Marktsituationen. (Schaaf, 2013) (Schraml, 2014)

Abhängigkeit

Durch den Bezug eines proprietären Softwareprodukts ist man vom Hersteller in vielerlei Hinsicht abhängig. Durch den unzugänglichen Source Code kann und darf man nicht selbst in das erhaltene Produkt eingreifen, und ist auf die Liefer- und Entwicklungszyklen des Herstellers angewiesen. Ebenso ist es möglich, dass man auf die Hardware des Herstellers angewiesen ist, und nicht auf Drittanbieter zurückgreifen kann. Verschwindet der Hersteller vom Markt, gibt es möglicherweise keinen Nachfolger, der Supportleistungen anbietet oder das Produkt weiterentwickelt. (Schaaf, 2013) (Schraml, 2014)

Limitierung von Ressourcen

Durch die Geheimhaltung des Source Codes ist der Hersteller auf die Ressourcen seines eigenen Unternehmens oder von Vertragspartnern limitiert. Entwicklerinnen und Entwickler sind daher in begrenzter Zahl für Wartung und Weiterentwicklung vorhanden und planbar. (Schaaf, 2013) (Storz, 2012)

Eingeschränkte Zielgruppe

Durch hohe Lizenz- und Wartungskosten ist die Zielgruppe des Herstellers auf Käufer limitiert, die diese Kosten auch aufbringen können und wollen. Handelt es sich um ein Nischenprodukt, kann bereits der (Zahlungs-)Ausfall weniger Kunden ein Risiko für den Hersteller darstellen. (Schaaf, 2013)

4.5 Fazit

Das Wesen eines proprietären Softwareprodukts liegt in der Geheimhaltung dessen Source Codes. Der Eigentümer bzw. die Eigentümerin des Softwareprodukts, welcher eine natürliche oder juristische Person ist, sieht darin ein schützenswertes Gut und möchte alleine über dessen Art und Weise der Verwertung entscheiden. Bei den meisten proprietären Softwareprodukten handelt es sich um kommerzielle Softwareprodukte, da der Hersteller durch die Geheimhaltung des Source Codes Softwarelizenzen verkaufen möchte, wodurch Gewinne erzielt werden sollen. Dieses Geschäftsmodell begann sich ab Ende der 1970er Jahre zu verbreiten, als Software von Hardware unabhängig wurde und sich der Computer zum Massenprodukt entwickelte.

Proprietäre Software wird über Lizenzen verkauft, welche die Einräumung von Nutzungsrechten, aber auch Pflichten für die Erwerberin bzw. den Erwerber bedeuten. Dessen Gestaltung bleibt dem Hersteller überlassen, und kann von Fall zu Fall unterschiedlich ausfallen.

Für Lizenzen gibt es eine Vielzahl von möglichen Modelltypen. So kann es sich dabei beispielsweise um nutzerbezogene, wertbezogene, zeitbezogene und infrastrukturbezogene Modelle handeln, welche auch kombiniert werden können. Ein Kopierschutz der Software soll dazu beitragen, dass das erworbene Produkt nicht unrechtmäßig vervielfältigt wird.

Bei einer Lizenz für ein proprietäres Softwareprodukt handelt es sich im Regelfall um eine sogenannte EULA, ein End User License Agreement. Dieses enthält die Rahmenbedingungen der Verwendung des Softwareprodukts, sowie die Bestimmungen des Urheberrechts. Eine Zustimmung zur EULA wird dadurch erreicht, dass der Nutzer ohne diese das Softwareprodukt nicht nutzen kann.

5 OPEN SOURCE SOFTWARE

Die zuwachsende Verbreitung von Open Source Softwareprodukten hat eine wachsende Zahl an rechtlichen Fragestellungen sowohl seitens der Hersteller als auch der Nutzerinnen und Nutzer zur Folge (Jaeger & Metzger, 2016). Das Wesen von Open Source Software sowie deren rechtliche Hintergründe werden daher wie folgt näher erläutert.

5.1 Was ist Open Source Software?

Im Gegensatz zu Geschäftsmodellen proprietärer Softwareprodukte ist der Source Code von Open Source Softwareprodukten frei. Mit frei ist hierbei gemeint, dass dieser der Allgemeinheit frei zugänglich ist, und diese den Source Code bearbeiten und weiterverbreiten kann. Dadurch profitiert die Gemeinschaft als Gesamtheit an Weiterentwicklungen anderer. (Weber, 2005)

Weist eine Softwarelizenz folgende Merkmale auf, werden diese laut (Bdeiwi, 2016) von der sogenannten Open Source Initiative als Open Source Software anerkannt:

- Die Software und ihr Source Code liegen in einer für den Menschen verständlichen und lesbaren Form vor.
- Die Software darf beliebig kopiert, genutzt und verbreitet werden.
- Die Software darf verändert und in veränderter Form weiterverbreitet werden.

Ergänzend der genannten Punkte werden von der Open Source Initiative (OSI, 2007) folgende Kriterien festgelegt:

- Niemand darf die freie Weiterverbreitung einschränken. Daher dürfen für Open Source Softwareprodukte keine Lizenzgebühren verlangt werden.
- Wird eine Bearbeitung weiterverbreitet, muss die neue Version dieselben Lizenzbedingungen aufweisen wie die vorhergehende Version. Es dürfen auch keine weiteren hinzugefügt werden.
- Bei einer Bearbeitung muss zum offenen Source Code auch ein Dokument zugänglich gemacht werden, aus welchem genau hervorgeht, welche Teile des Source Codes bearbeitet wurden.
- Weder bestimmte Personen bzw. Gruppen, noch die Nutzeranzahl dürfen vom Gebrauch des Open Source Softwareprodukts ausgeschlossen bzw. eingeschränkt sein. Auch das Open Source Softwareprodukt selbst darf keine anderen Softwareprodukte einschränken.
- Werden einzelne Open Source Softwareprodukte aus einem Softwarepaket einzeln weiterverbreitet, so gelten sowohl für die einzelnen Softwareprodukte als auch für das Softwarepaket dieselben Lizenzbedingungen.

Die genaue Definition laut (OSI, 2007) findet sich in Anhang A.

Die Open Source Initiative stellt dabei eine gemeinnützige Vereinigung dar, welche sich für eine einheitliche und nicht ideologische Definition von Open Source Software einsetzt. (Schaaf, 2013)

(Bruegge, et al., 2004) nennt folgende vier Besonderheiten von Open Source Softwareprodukten:

- die Softwarelizenz,
- die nicht-kommerzielle Einstellung,
- den hohen Grad an gemeinschaftlicher Entwicklungsleistung und
- den hohen Grad an räumlicher Verteilung der Softwareentwicklerinnen und Softwareentwickler.

Jedoch ist alleinig das Aufweisen einer der laut OSI anerkannten Open Source Softwarelizenz ausschlaggebend dafür, ob es sich um ein Open Source Softwareprodukt handelt oder nicht. (Bruegge, et al., 2004)

Um Open Source Softwareprodukte klar abgrenzen zu können, ist zu erwähnen, dass Public Domain Software, sowie Shareware und Freeware im Allgemeinen nicht zu den Open Source Softwareprodukten gehören, da diese auf unterschiedliche Weisen den genannten Kriterien widersprechen. (Hennig, 2009)

Wie In Kapitel 2.3 besprochen weist Public Domain Software keinen oder in Teilen keinen Urheberrechtsschutz auf und stellt für die Nutzerin bzw. den Nutzer ein einfaches Nutzungsrecht dar. Da der Source Code von Public Domain Software nicht frei zugänglich sein muss, kann diese nicht den Open Source Softwareprodukten zugeordnet werden. Auch umgekehrt kann Open Source Software nicht Public Domain Software zugeordnet werden, da der Nutzerin bzw. dem Nutzer erlaubt wird, die Software zu verändern und zu verwerten. Bei Shareware handelt es sich um eine besondere Vermarktungsstrategie für kommerzielle Softwareprodukte.

Die Free Software Foundation setzt sich für freiheitsgewährende Software ein, womit genauer erlaubt ist, die Software

- frei zu ändern und zu untersuchen,
- weiterzugeben, und
- für beliebige Zwecke zu verwenden.

Diese Merkmale sind bei proprietärer Software nicht gegeben, da diese mittels EULA entzogen werden. (FreeSoftwareFoundation, 2016)

Außerdem verdeutlicht die Unterscheidung freier und unfreier Softwarekategorien nochmals, dass es sich bei dem Begriff „frei“ in Bezug auf Open Source Softwareprodukte in erster Linie um die Verfügbarkeit des Source Codes handelt, welcher in selbiger oder bearbeiteter Form kostenlos oder kostenpflichtig weitergegeben werden darf. Es ist also durchaus möglich, dass ein Open Source Softwareprodukt kostenpflichtig vermarktet wird, und ebenso, dass ein proprietäres Softwareprodukt kostenlos zur Verfügung gestellt wird. Da proprietäre Softwareprodukte mit dem Begriff „frei“ im Sinne von kostenfrei werben können, bedarf es daher bei einem „freien“

Softwareprodukt immer einer genauen Prüfung, um welches „frei“ es sich tatsächlich handelt. (FreeSoftwareFoundation, 2016)

Nicht das Software Artefakt für sich steht laut Aussage von (Weber, 2005) im Vordergrund, sondern der gesamte Gestaltungs- und Entstehungsrahmen um das Softwareprodukt herum. Um verstehen zu können, worum es sich bei dem Open Source Prozess handelt, muss man sich mit der Frage auseinandersetzen, welches Problem durch Open Source gelöst werden soll, und sich dadurch den Menschen widmen, die an diesem Prozess teilhaben. Bei Nutzerinnen und Nutzern von Open Source Softwareprodukten handelt es sich nicht ausschließlich um reine Konsumenten. Vielmehr können sie aktiv an einer Gemeinschaft teilhaben, sich unter einander austauschen und in unterschiedlicher, individueller Art und Weise in den Entwicklungsprozess integrieren. (Weber, 2005)

(Weber, 2005) beschreibt das Recht die Software weiterverbreiten zu dürfen als fundamentale Eigenschaft von Open Source Software. Bei der Community handelt es sich um Softwareentwicklerinnen und Softwareentwickler, die über geografische, kulturelle, sprachliche und soziale Grenzen hinweg gemeinsam an der Weiterentwicklung eines Softwareprodukts arbeiten. Dabei verzichten diese auf Firmenstrukturen und stellen das kollaborative Ziel, hochwertige, komplexe Software zu entwickeln, in den Vordergrund.

Einige Gründe für den Erfolg von Open Source Software aus Sicht der Softwareentwicklerinnen und Softwareentwickler sind laut (Bruegge, et al., 2004) und (Faber, 2008):

- Das Entwickeln eines optimal angepassten Softwareprodukts,
- Wertschätzung der erbrachten Leistung,
- Austausch und Zusammenarbeit mit anderen Softwareentwicklerinnen und Softwareentwicklern,
- Reputationsmöglichkeit innerhalb der Community,
- Erweiterung der eigenen Fähigkeiten,
- Der Spaß am Programmieren selbst,
- Verbesserte Marktsituation des Softwareprodukts durch OSI Lizenzierung

Dabei wird deutlich, dass soziale Faktoren das Fundament und die Motivation der Community bilden, und nicht finanzielle. (Faber, 2008)

Wirtschaftliches Interesse im Sinne des Erzielens von Gewinnen liegt bei der Entwicklung von Open Source Software nicht im Vordergrund, da originär auf einen betriebswirtschaftlichen Rahmen verzichtet wird. Auch können keine Gewinne direkt durch den Verkauf von Lizenzen lukriert werden (Weber, 2005). Unternehmen haben jedoch mit der historischen Entwicklung von Open Source Software Wege gefunden, mittels gezielter Weiterentwicklung und Vermarktung Profit zu generieren. (Faber, 2008)

5.2 Historischer Überblick

Open Source Software ist ein Ergebnis einer immer stärker miteinander vernetzten Welt. Sowohl die Existenz proprietärer Software als auch die Entwicklung des Internets sind eng mit der Entstehung von Open Source Software verbunden. (Jaeger & Metzger, 2016)

Obwohl die Geschichte von Open Source Software im Sinne einer frei veränderbaren und nutzbaren Software bis in die 60er Jahre des zwanzigsten Jahrhunderts zurückgeht, konnte Open Source Software erst mit der Erfolgsgeschichte von Linux in den neunziger Jahren auch im öffentlichen und privaten Wirtschaftssektor Fuß fassen. Seitdem stellt sich für immer mehr Unternehmen die Frage, ob sich der Einsatz von Open Source Softwareprodukten bzw. der Umstieg von proprietären Softwareprodukten zu Open Source Softwareprodukten für sie lohnt. (Hennig, 2009)

Wie (Landley, 2009) in Kapitel 5.2, beschreibt auch (Jaeger & Metzger, 2016) die Zeit der 60er und 70er Jahre, zu welcher Hardware und Software noch nicht getrennt voneinander betrachtet wurden. Aufgrund der starken Spezialisierung und dem Fokus auf Hardware, war der Source Code der zugehörigen Software der Nutzerin bzw. dem Nutzer frei zugänglich oder wurde von ihr bzw. ihm selbständig oder in Kooperation mit dem Hersteller der Hardware wie IBM entwickelt. Dadurch entstand ein Netzwerk an Softwareentwicklerinnen und Softwareentwicklern, worin deren Erfahrungen und Weiterentwicklungen miteinander geteilt werden konnten. Zu dieser Zeit handelte es sich dabei um die Produktion von Großrechnern, welche von der Firma IBM hergestellt wurden. Der gebündelte Vertrieb von Hardware und Software wurde dabei als ein wichtiger Faktor der Monopolstellung von IBM gesehen, da es kaum Wettbewerb in dieser Branche gab. Daher wurde im Jahre 1969 von amerikanischen Behörden ein Kartellverfahren veranlasst, welches die separate Betrachtung von Hardware und Software zur Folge hatte. Dies stellte aber erst mit der Entwicklung von Mikroprozessoren und der Hervorbringung des ersten Computers von IBM im Jahre 1981 die Grundlage der heutigen Softwareindustrie dar, da der Computer nun der Masse zugänglich war.

5.2.1 Das Unix Betriebssystem

In der Geschichte der Open Source Software spielt auch das Betriebssystem **Unix** eine Rolle. Dieses wurde 1969 von Ken Thompson entwickelt, welcher bei der Telefongesellschaft AT&T tätig war. In nur vier Wochen erstellte Thompson das Grundgerüst des Betriebssystems. Das Ziel, ein multitaskingfähiges System zu entwickeln, setzte er zusammen mit Dennis Ritchie um. Das entstandene Betriebssystem Unix wurde in weiterer Folge auch intern bei AT&T eingesetzt. Unix erlangte an Beliebtheit bei Universitäten, nicht zuletzt aufgrund der Tatsache, dass AT&T aus kartellrechtlichen Gründen nicht im herkömmlichen Sinne kommerziell vermarkten durfte. Die Lizenzkosten für Unix waren daher gering. Des Weiteren spielte die leichte Portierbarkeit des Betriebssystems eine wichtige Rolle. Da es auf die Programmiersprache „C“ umgeschrieben wurde, konnten aufgrund der Bekanntheit der Programmiersprache, Entwicklerinnen und Entwickler das Betriebssystem weiterentwickeln. Unix wurde in weiterer Folge auch aufgrund fehlenden Supports durch AT&T an den Universitäten weiterentwickelt. Beliebtheit erlangte das

1977 veröffentlichte BSD-Unix, welches von der Berkeley Universität herausgebracht wurde. Als Vorläufer des Internets gilt „Usenet“, worüber es Softwareentwicklerinnen und Softwareentwicklern möglich war, innerhalb der Universitäten miteinander zu kommunizieren. Durch den Anschluss an das „ARPANET“ gelang die Kommunikation auch zwischen den Universitäten. 1984 stellten erneut kartellrechtliche Gründe eine Wende in der Geschichte von Unix dar, da AT&T in mehrere kleinere Unternehmen zerschlagen wurde. Für das BSD-Unix, welches Source Code von AT&T enthielt, waren bereits seit 1975 Lizenzgebühren zu entrichten. Diese stiegen aufgrund der neu erlangten Wettbewerbsfähigkeit von AT&T rapide an. Daraufhin begann man den BSD Source Code um den Source Code von AT&T zu bereinigen, woraus in weiterer Folge neue BSD-Projekte entstanden. (Jaeger & Metzger, 2016) (Weber, 2005)

5.2.2 Die Free Software Foundation

Als Richard Stallman 1980 am MIT feststellen musste, dass er die Programmierung eines Druckers aufgrund dessen proprietärer Treibersoftware nicht mehr modifizieren konnte, begann sich dieser für freie Software einzusetzen (Williams, 2002). Da er den zahlreichen Restriktionen, die proprietäre Softwareprodukte mit sich bringen, die Stirn bieten wollte, gründete er 1984 das **GNU** Projekt. Ziel war es, ein Betriebssystem zu entwickeln, dass Unix-kompatibel ist, aber nicht den Bedingungen von AT&T unterliegt. Die Benennung zeigt seine Art des Humors, da GNU ein rekursives Akronym für „GNU´s not Unix“ ist. 1985 gründete Stallman die Free Software Foundation (FSF), um das Wesen freier Software zu manifestieren (Williams, 2002). Dies unterstrich Stallman 1989 mit der Entwicklung der GNU General Public License (GPL), welche die Basis für den Großteil der innerhalb des GNU Projektes entwickelten Softwareprodukte darstellte. Dafür kündigte Stallman sogar am MIT, da er nicht wollte, dass seine Entwicklungen als Arbeitsleistung im Namen der Universität gelten. (Jaeger & Metzger, 2016) (Weber, 2005)

5.2.3 Das Linux Betriebssystem

1991 begann Linus Torvalds unabhängig von Stallman ein eigenes Betriebssystem zu entwickeln. Die Basis dafür stellte ein neu erschienener Computer, auf dessen 386-Chip ein Unix-Derivat namens Minix lief. Da Torvalds über Minix allerdings nicht auf Unix-basierende Universitätsrechner zugreifen konnte, entwickelte er diese und weitere Funktionen eigenständig. Dies stellte das Grundgerüst des Betriebssystems **Linux** dar, an dessen Weiterentwicklung sich nun viele weitere Softwareentwickelrinnen und Softwareentwickler beteiligten. Torvalds stellte Linux 1992 unter die GPL V2, um den freien Zugang zu Linux gewährleisten zu können. 1994 wurde Version 1 von Linux veröffentlicht, dessen Erfolg unter anderem auf das Netzwerk der Softwareentwicklerinnen und Softwareentwickler, die Verbreitungsmöglichkeit über das Internet und die Verwendung von bestehenden GNU-Komponenten zurückzuführen ist. Daher stammt auch die Bezeichnung „GNU/Linux“, unter welcher das Linux-Betriebssystem auch bekannt ist. Heute beteiligen sich mehrere tausend Softwareentwicklerinnen und Softwareentwickler und mehrere hundert Unternehmen an der kollaborativen Weiterentwicklung von Linux. Auch Linus Torvalds ist als organisatorischer Leiter des Entwicklungsprozesses für den Standardkernel nach

wie vor am Linux-Projekt beteiligt, und wird von der 2007 gegründeten Linux Foundation finanziert. (Jaeger & Metzger, 2016)

5.2.4 Die Open Source Initiative

Erklärtes ideologisches Ziel der Free Software Foundation war und ist der herstellerunabhängige Umgang mit Software, und nicht deren Vermarktungsmöglichkeiten. Der Begriff „Free Software“ stellte allerdings seit dessen Existenz ein Problem dar, da die Bedeutung des enthaltenen „Free“ nicht genügend Aussagekraft besaß. Der Gedanke der freien Zugänglichkeit und des freien Wesens wurde mit frei im Sinne von kostenlos oder besitzerlos vermischt. Um der Softwareindustrie die Berührungsängste mit freier Software zu nehmen, entschieden sich einige Protagonisten dazu, Maßnahmen zu ergreifen. Als Bruce Berens, Eric S. Raymond und Tim O`Reilly sich am 03.02.1998 in Palo Alto, Kalifornien, trafen, um sich über Alternativen zum Begriff „Free Software“ zu besprechen, endete dies mit der Bezeichnung „Open Source“ und der Gründung der Open Source Initiative. Dieser Schritt traf auf großen Zuspruch einiger namhafter Softwarehersteller wie IBM und Oracle, welche ankündigten, dieses Konzept aufgreifen zu wollen. Stallmann, welcher mit der Free Software Foundation ein idealistisches, gesellschaftliches Modell verfolgte, wurde von den Gründern der Open Source Initiative bewusst nicht zu besagtem Treffen eingeladen. Diese verfolgten im Gegensatz zu Stallmann den Ansatz, Open Source als Entwicklungsmodell für gute Software zu sehen, wofür der freie Zugang zum Source Code der Software ein notwendiges Mittel ist. Stallman gibt zwar an, dass sich beide Vereinigungen in deren ausgesprochenen Empfehlungen einig sind, diese dennoch aufgrund des unterschiedlichen Grundgedanken als separat voneinander zu betrachten sind. Der Konflikt der beiden Parteien fußt in Stallmans Befürchtung, offener Source Code werde zum ausschlaggebenden Kriterium für die Bezeichnung des Open Source Softwareprodukts, entgegen der Grundsätze der Free Software Foundation (Jaeger & Metzger, 2016). (Schaaf, 2013)

5.2.5 Open Source bis heute

Anknüpfend an die Erfolgsgeschichte von Linux entwickelten sich im Laufe der Jahre zahlreiche darauf aufbauende aber auch unabhängige Open Source Softwareprojekte. Einerseits wurden neue Funktionen entwickelt, die in der Welt der Open Source Software bislang fehlten, andererseits wurde das Ziel verfolgt, Alternativen zu existierender proprietärer Software zu bieten. Bedeutsame Vertreter dieser Entwicklungen sind auszugsweise der Apache-Webserver, das Datenbanksystem MySQL, die Entwicklungsumgebung Eclipse, das Instant Messaging Programm Jabber sowie das Betriebssystem Android (Bruegge, et al., 2004) (Jaeger & Metzger, 2016). Unterstützung findet die Bewegung zu Open Source Softwareprodukten auch durch namhafte Vertreter der Softwareindustrie wie etwa IBM, HP oder Google, welche darin unter anderem technologische oder wettbewerbsorientierte Vorteile sehen. Auch von öffentlicher Seite findet die Bewegung der Open Source Software Anerkennung, wobei beispielsweise die Stadt München 2004 mit dem Umstieg auf Linux den Grundstein dafür innerhalb der Europäischen Union legte. Diese wiederum brachte die European Union Public License (EUPL) heraus, sowie

einen Leitfaden, welcher der Beschaffung von Open Source Software durch die öffentliche Hand dienlich sein soll. (Jaeger & Metzger, 2016)

In Anlehnung an (Schaaf, 2013) stellt sich folgender Auszug aus der chronologischen Entwicklung von Open Source Software dar:

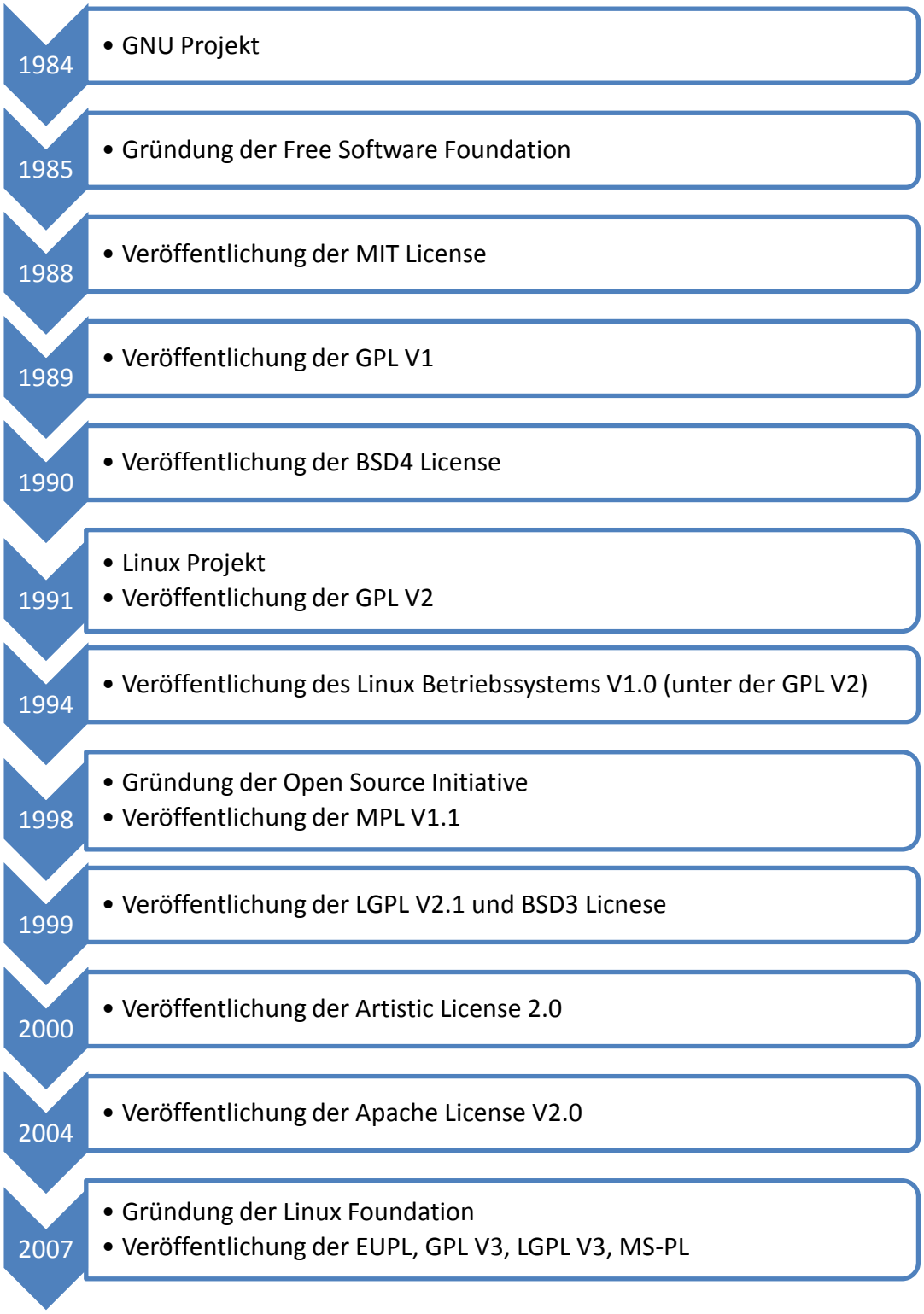

Abbildung 3 - Chronologische Entwicklung von Open Source

Darauf aufbauend entwickelten sich zahlreiche Open Source Softwareprojekte, zu denen bekannte Vertreter wie Samba, Android, Google Chrome, Bootstrap und Git zählen. (Schrape, 2015)

5.3 Open Source Softwarelizenzen

Da Open Source Softwareprodukte dem Urheberrechtsgesetz unterliegen, ist für deren Verwendung eine Lizenz gemäß dem Urheberrechtsgesetz von Nöten. (Bdeiwi, 2016)

Die sogenannte Copyleft Klausel in einer Open Source Lizenz gibt Auskunft über die Lizenzänderungsmöglichkeit bei Bearbeitung und weiterer Distribution eines Open Source Softwareprodukts. Die in Anlehnung an das Copyright von proprietären Produkten existierende Klausel ist allerdings in unterschiedlichen Ausprägungen vorhanden. (Hennig, 2009)

Die Entscheidung darüber, ob eine Lizenz als Open Source Lizenz anzusehen ist, liegt bei der Open Source Initiative. Dabei muss die Lizenz in jedem Fall der Definition von Open Source laut Open Source Initiative entsprechen. (Schaaf, 2013)

Nach (Bruegge, et al., 2004) verteilte sich die Nutzung verschiedener Open Source Softwarelizenzen folgendermaßen:

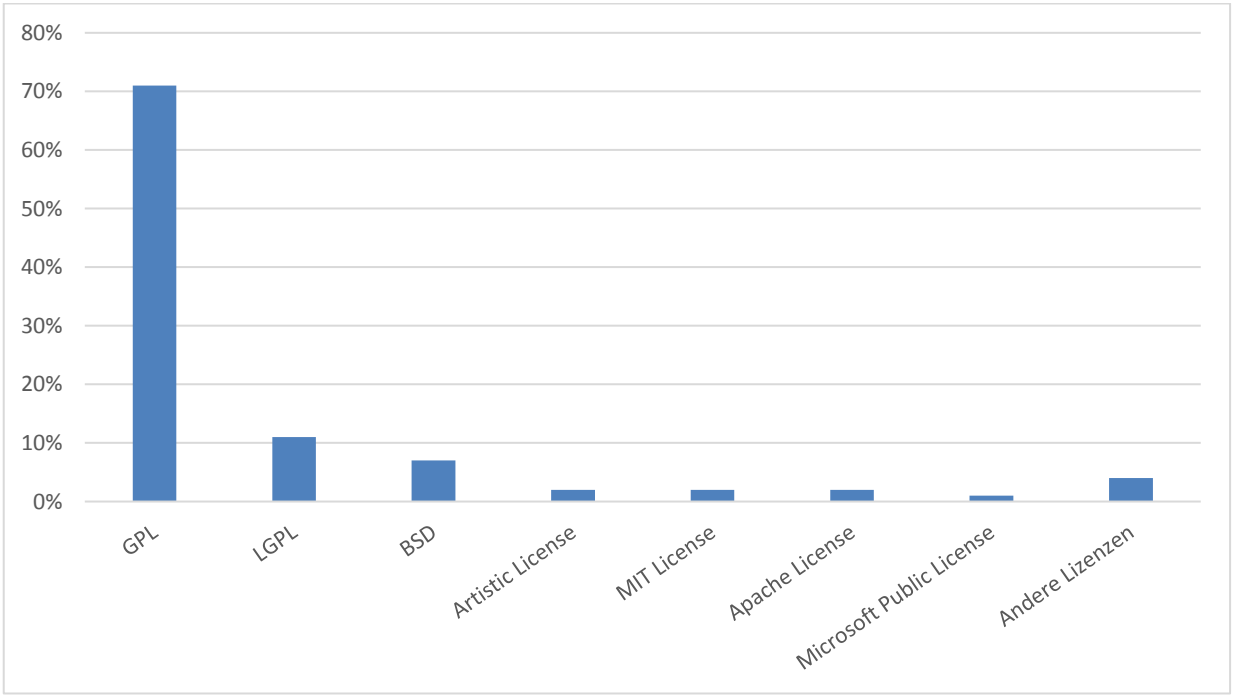

Abbildung 4 - Verteilung von Open Source Lizenzen nach (Brügge, et al., 2004)

Neun Jahre später ergab sich nach (Schaaf, 2013) folgende Verteilung der bekannten Open Source Softwarelizenzen:

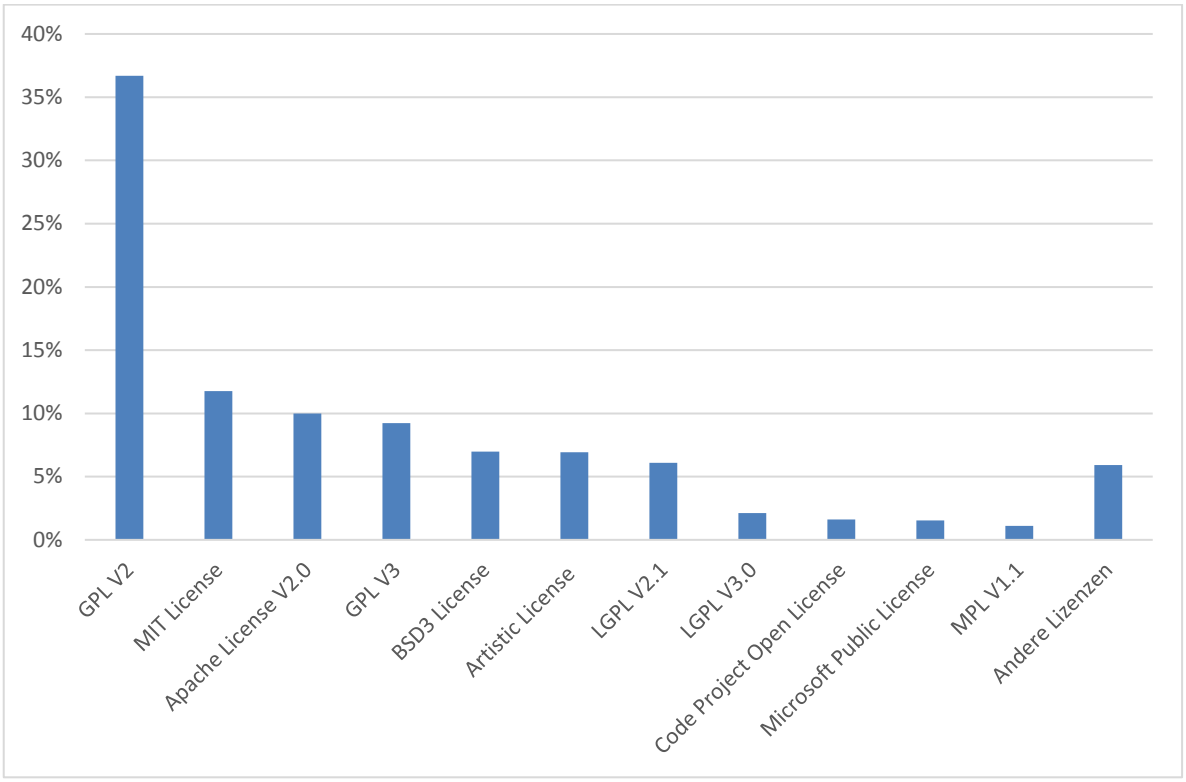

Abbildung 5 - Verteilung von Open Source Lizenzen nach (Schaaf, 2013)

(Wilson, 2015) veröffentlicht zwei Jahre später folgende Statistik:

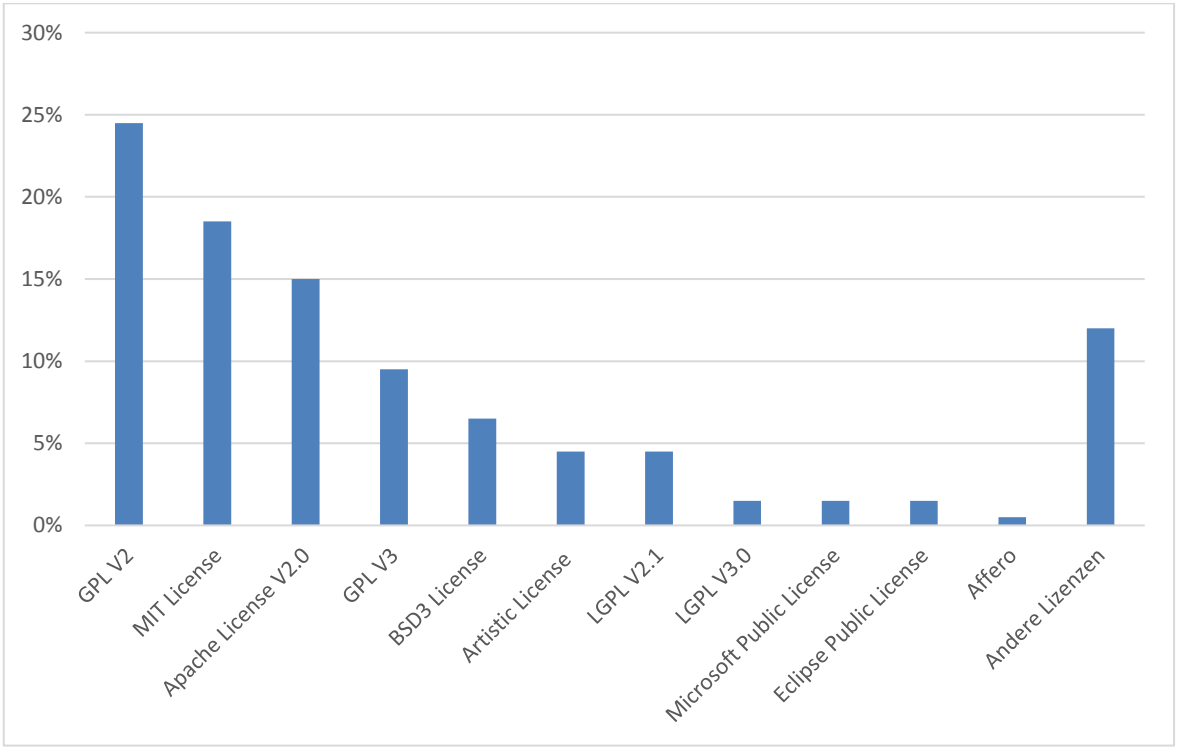

Abbildung 6 - Verteilung von Open Source Lizenzen nach (Wilson, 2015)

Beide zuvor genannten Autoren berufen sich dabei auf die damals aktuellen Daten des Unternehmens Black Duck, wobei (BlackDuck, 2016) selbst folgende aktuellste Statistik herausgibt:

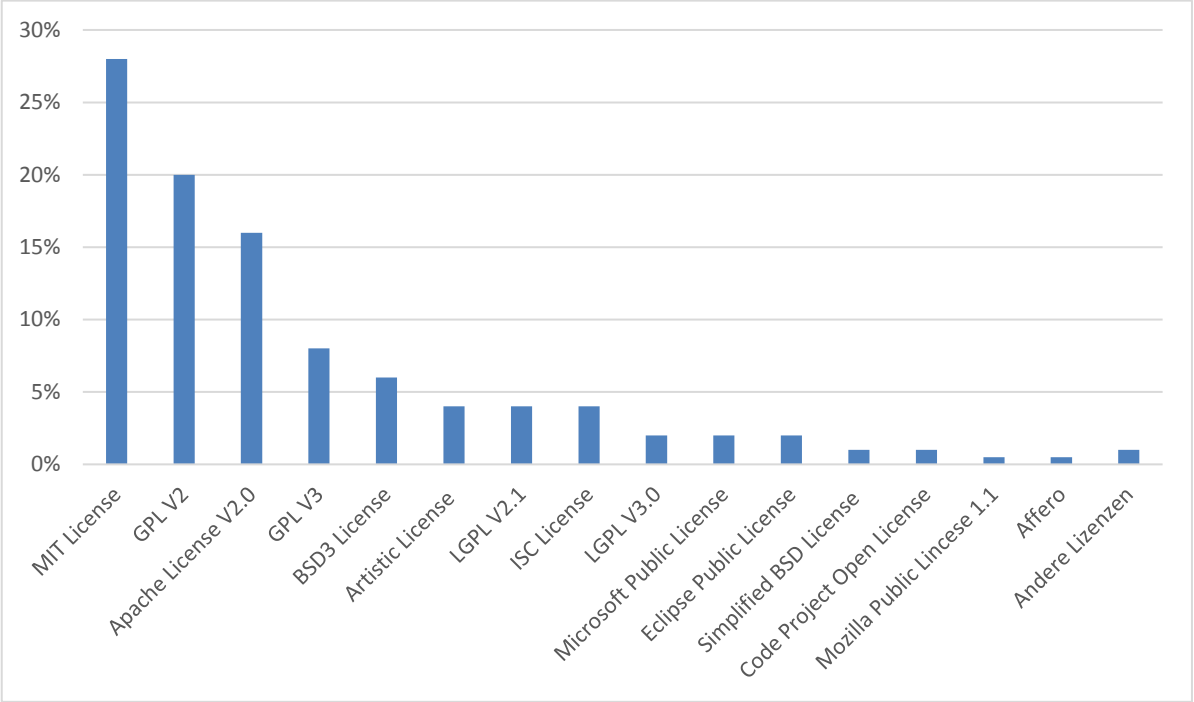

Abbildung 7 - Verteilung von Open Source Lizenzen nach (BlackDuck, 2016)

Wie man anhand dieses Verlaufes sehen kann, wurde die GPL Lizenz, welche jahrelang an erster Stelle hinsichtlich der Häufigkeit ihrer Verwendung lag, mittlerweile von der MIT Lizenz überholt. In der Praxis hat sich durchgesetzt, die Open Source Lizenzen nach der Art der Copyleft Klausel zu kategorisieren (Schaaf, 2013). Für die in Abbildung 7 gelisteten Open Source Lizenzen sieht dies in Anlehnung an (Schaaf, 2013) wie folgt aus:

Open Source Lizenz	**Implementierung des Copyleft**
MIT License	Kein Copyleft
GPL V2	Strenges Copyleft
Apache License V2.0	Kein Copyleft
GPL V3	Strenges Copyleft
BSD3 License	Kein Copyleft
Artistic License	Wahlrecht
LGPL V2.1	Beschränktes Copyleft
ISC License	Kein Copyleft
LGPL V3.0	Kein Copyleft
Microsoft Public License	Kein Copyleft
Simplified BSD License	Kein Copyleft
Code Project Open License	Kein Copyleft
Mozilla Public License 1.1	Beschränktes Copyleft
Affero 3.0	Strenges Copyleft

Tabelle 5 - Implementierung des Copylefts für häufig genutzte Open Source Lizenzen

Die gelisteten Lizenzen können dabei als Vorlage für einen potentiellen Open Source Softwarehersteller dienen, um diese als Basis für die Vertragsgestaltung heranzuziehen. Dennoch gibt es über 200 verschiedene Open Source Lizenzen, da Hersteller von neuen Softwareprodukten für diese mitunter auch neue Lizenzen entwickeln, anstatt von bestehenden Gebrauch zu machen. (Schaaf, 2013)

5.3.1 Lizenzen mit strengem Copyleft

Lizenzen mit strenger Copyleft Klausel sind laut (Jaeger & Metzger, 2016) und (Schaaf, 2013) unter anderem folgende:

- GNU General Public License, Version 2 (GPL-2.0)
- GNU General Public License, Version 3 (GPL-3.0)
- GNU Affero General Public License, Version 3 (AGPL-3.0)
- Eclipse Public License (EPL)
- European Union Public License (EUPL)

Aufgrund der Namen der stärksten Vertreter werden Lizenzen dieser Gattung auch GPL-artige Lizenzen genannt. (Schaaf, 2013)

Ein strenges Copyleft hat zur Folge, dass jegliche Weiterverbreitung einer Bearbeitung eines Open Source Softwareprodukts mit denselben Lizenzbedingungen zu erfolgen hat, wie das ursprüngliche Open Source Softwareprodukt. Damit wird sichergestellt, dass jegliche weiterentwickelte und weiterverbreitete Version des Open Source Softwareprodukt im Wesen unverändert, und somit frei bleibt. Dies trifft auch wie gelistet auf die GPL Lizenz zu, bei welcher es sich um die meist verbreitete Open Source Softwarelizenz mit handelt. (Jaeger & Metzger, 2016) Dieser Aussage widerspricht jedoch (BlackDuck, 2016), welche die MIT Lizenz als mittlerweile häufigste verwendete Open Source Lizenz listet.

Dennoch ist die GNU General Public License (GPL) häufigster Vertreter der GPL-artigen Lizenzen. Im Wesentlichen erstellt durch Richard Stallmann, welcher unter anderem als Gründer des GNU Projektes bekannt ist, ist die GPL die erste Open Source Lizenz, die eine Copyleft Klausel aufweist. Version 1 ist heute nicht mehr von Relevanz. Die 1991 verfasste Version 2 jedoch ist immer noch verbreiteter als die 2007 herausgebrachte Version 3 der GPL. (Schaaf, 2013)

Die Free Software Foundation ist alleiniger Herausgeber neuer GPL Versionen. Einer Nutzerin bzw. einem Nutzer eines Softwareprodukts, welches unter einer GPL Lizenz steht, werden durch das strenge Copyleft Rechte und Pflichten auferlegt, wobei die Pflichten erst eintreten, wenn die Software über die gewöhnliche Nutzung hinaus verwendet wird. Dies wird durch die Einräumung eines gewöhnlichen Nutzungsrechts erlangt, wobei für die Entwicklerinnen und Entwickler der Software eine Haftung ausgeschlossen wird. Weiterentwicklungen müssen durch das strenge Copyleft unter derselben GPL Lizenz stehen, um allen Menschen freien Zugang daran sicherzustellen. Obwohl keine Lizenzkosten erhoben werden dürfen, ist es durchaus möglich,

dass etwa Gebühren für Vertriebswege vom Hersteller der Bearbeitung erhoben werden. (Schaaf, 2013)

5.3.2 Lizenzen mit beschränktem Copyleft

Lizenzen mit beschränkter Copyleft Klausel sind laut (Jaeger & Metzger, 2016) unter anderem folgende:

- Mozilla Public License, Version 2 (MPL-2.0)
- GNU Lesser General Public License (LGPL)
- Microsoft Public License (MS-PL)

Aufgrund des Namens des stärksten Vertreters werden Lizenzen dieser Gattung auch MPL-artige Lizenzen genannt. (Schaaf, 2013)

Während bei Lizenzen mit strengem Copyleft verhindert wird, dass Bearbeitungen des Open Source Softwareprodukts unter anderen Lizenzbedingungen freigegeben werden können, ist dies bei Lizenzen wie beispielsweise der **Lesser General Public License (LGPL)** mit beschränkter Copyleft Klausel möglich. Genauer ist dies möglich, indem die ursprüngliche freie Software mit Bearbeitungen unter anderen Lizenzbedingungen kombiniert werden kann. (Jaeger & Metzger, 2016) Dabei ist jedenfalls zu gewährleisten, dass die Modifikation bzw. der eigens erstellte Source Code in einer eigenen Datei gespeichert wird, welche in weiterer Folge eine eigene Lizenz aufweisen darf (Schaaf, 2013).

Software, die auf Bibliotheken einer LGPL lizenzierten Software zugreift, muss bei MPL-artigen Lizenzen nicht unter derselben Lizenz stehen und kann daher auch kommerziell sein. Bei GPL wäre dies nicht möglich, da die Software durch den Zugriff auf eine Bibliothek unter der GPL auch unter dieser Lizenz verbreitet werden müsste. (Schaaf, 2013)

Genauer definiert dies die Version 2.1 der LGPL. Diese legt fest, dass Softwareprodukte, die auf Bibliotheken unter einer LGPL zugreifen, eine andere Lizenz aufweisen dürfen, wenn die Bibliothek unter der LGPL dabei nicht Bestandteil der vertriebenen Software ist. Ist die Bibliothek unter der LGPL allerdings Teil des vermarkteten Softwareprodukts, muss diese auch unter der LGPL lizenziert werden. Hierzu gibt es in der Version 2.1 eine in der Praxis umstrittene Ausnahmeregelung, im Falle, dass die Nutzerin bzw. der Nutzer etwa Rechte zur Anpassung oder zur Fehlerbehebung der Software hat. Die Nutzerin bzw. der Nutzer muss dabei aber jedenfalls sicherstellen, dass im weiterentwickelten Softwareprodukt auf unter LGPL stehende Bibliotheken hingewiesen wird, und dieses auch mit einer adaptierten Version der Bibliothek ausführbar ist. (Schaaf, 2013)

Version 3 der LGPL wurde 2007 im Rahmen der neu herausgegebenen GPL mitdefiniert und beschreibt lediglich Ausnahmen zur GPL um Redundanzen zu vermeiden. Durch diesen Umstand gelten für Softwareprodukte, die unter der LGPL V3 lizenziert werden, sowohl die Richtlinien der GPL V3 als auch die der LGPL V3. (Schaaf, 2013)

Die **Microsoft Public License (MS-PL)** wurde ursprünglich als Microsoft Permissive License im Jahre 2005 von Microsoft herausgebraucht. 2007 wurde die zu Microsoft Publice License umbenannte Lizenz von der Open Source Initiative anerkannt. Bemerkenswert hierbei ist, dass die MS-PL aus Microsofts „Shared Source Initiative“ entstand, welche zu Beginn als Gegenbewegung zur Open Source Initiative angesehen werden konnte. (Jaeger & Metzger, 2016)

Die **Mozilla Public License (MPL)** wurde im Rahmen des Netscape Communicator Projektes entwickelt und bewirkt, dass nur Bearbeitungen ursprünglicher Dateien des betreffenden Softwareprodukts den Bestimmungen des Copylefts unterliegen. Softwareprodukte des Mozilla Projektes werden mittlerweile dual lizenziert, also unter verschiedenen Lizenzen angeboten. Daher hat die MPL heute an Bedeutung verloren. (Jaeger & Metzger, 2016)

5.3.3 Lizenzen ohne Copyleft

Lizenzen ohne Copyleft Klausel sind laut (Jaeger & Metzger, 2016) unter anderem folgende:

- Berkeley Software Distribution BSD
- Apache License
- MIT License

Aufgrund des Namens des stärksten Vertreters werden Lizenzen dieser Gattung auch BSD-artige Lizenzen genannt. (Schaaf, 2013)

Lizenzen ohne Copyleft Klausel weisen dem Namen nach keine Schutzbedingungen wie die beiden zuvor erwähnten Lizenzen auf. Dies hat zur Folge, dass eine Lizenznehmerin bzw. ein Lizenznehmer jegliche Weiterentwicklung in unfreie Eigenentwicklungen einbinden kann. Des Weiteren darf die Lizenznehmerin bzw. der Lizenznehmer eine Bearbeitung eines Open Source Softwareprodukts, welche unter einer Lizenz ohne Copyleft Klausel steht, proprietär weiterverbreiten. (Jaeger & Metzger, 2016) Die Lizenznehmerin bzw. der Lizenznehmer hat freie Hand darüber, unter welcher Lizenz bzw. unter welchen Lizenzen sie bzw. er das Softwareprodukt weiterverbreitet. Dabei kann es sich sowohl um Open Source Lizenzen handeln als auch um proprietären Vertrieb. (Schaaf, 2013)

Die **Berkeley Software Distribution BSD** Lizenz wurde an der Berkeley Universität entwickelt und ist als eine der liberalsten Open Source Lizenzen bekannt. Den Nutzerinnen und Nutzern wird mittels der Lizenz erlaubt, diese zu modifizieren und entsprechende Weiterentwicklungen zu verbreiten. Hierbei muss aber lediglich der Lizenztext des ursprünglichen Softwareprodukts mitverbreitet werden, eine Offenlegung des Source Codes ist nicht verpflichtend. Die BSD Lizenz verfolgt damit hauptsächlich das Ziel, den Haftungsausschluss für Softwareentwicklerinnen und Softwareentwickler sicherzustellen. Diese wiederum müssen verpflichtend durch den Lizenztext der BSD vollständig in jeder Bearbeitung eines Softwareprodukts unter der BSD Lizenz genannt werden. Die verschiedenen Versionen der BSD unterscheiden sich hauptsächlich darin, ob mit den genannten Softwareentwicklerinnen und Softwareentwicklern geworben werden darf, und ob auf den Namen der Berkeley Universität hingewiesen werden muss. (Schaaf, 2013)

Die **Apache Lizenz** entstand im Rahmen des Apache Projektes, wobei alle Softwareprodukte die innerhalb dieses Projektes entstanden, unter einer Apache Lizenz lizenziert wurde. Die Version 2.0 der Apache Lizenz ist kompatibel mit der Version 3 der GPL, was zur Folge hat, dass Source Code, welcher unter einer dieser Lizenzen entstanden ist, miteinander kombiniert werden kann. Wird dies vorgenommen, muss das entstandene Produkt allerdings gesamt unter der GPL Version 3 lizenziert werden. (Rouse, searchenterprisesoftware.de, 2016)

Die **MIT Lizenz** verdankt ihren Namen dem Massachusetts Institute of Technology, an welchem sie entwickelt wurde. Sie ist ähnlich der BSD, wobei der Unterschied darin liegt, dass manche BSD Lizenzen verbieten, mit dem Namen der Rechteinhaberin bzw. des Rechteinhabers zu werben. (Rouse, whatis.techtarget.com, 2011)

5.3.4 Lizenzen mit Wahlmöglichkeit

Lizenzen mit Wahlmöglichkeit sind laut (Jaeger & Metzger, 2016) unter anderem folgende:

- (Perl) Artistic License
- Clarified Artistic License
- Artistic License 2.0

Diese genannten Lizenzen stellen eine eigene Kategorie dar, da sich je nach Art und Weise der Bearbeitung, unterschiedliche Folgebedingungen und dem Namen nach verschiedene Wahlmöglichkeiten für die Nutzerin bzw. den Nutzer mit sich bringen. Hier bedarf es einer genauen Prüfung der in der Lizenz genannten Rechte und Pflichten durch die Nutzerin bzw. den Nutzer, um einen rechtmäßigen Umgang mit Bearbeitungen des Softwareprodukts zu gewährleisten. (Jaeger & Metzger, 2016)

Da die **Artistic License** die Grundlage für Perl in lizenzrechtlicher Hinsicht darstellt, wird sie oft auch mit dieser Bezeichnung in einem Zug genannt. Die Lizenzbedingungen der Artistic License erlauben die Verbreitung von Bearbeitungen unter anderen Lizenzen, jedoch müssen Dateien anders benannt und entsprechend dokumentiert werden. Des Weiteren dürfen lediglich Gebühren für den Vertrieb der Software in Rechnung gestellt werden, jedoch keine Lizenzgebühren an sich. Im Gegensatz zur Version 1 der Artistic License, welche nur von der Open Source Initiative anerkannt wurde, fand deren Überarbeitung, welche als Artistic License 2.0 bekannt ist, auch von der Free Software Foundation Anerkennung. (Schaaf, 2013)

5.3.5 Lizenzen mit Sonderrechten

Zu Lizenzen mit Sonderrechten zählen laut (Jaeger & Metzger, 2016) folgende:

- Netscape Public License (NPL)
- Q Public License (QPL)
- Apple Public License, Version 1.2 (APSL)

Bei diesen Lizenzen liegt der Fokus darauf, dass Unternehmen mittels Einräumung von Sonderrechten die Rechte an Weiterentwicklungen durch die Community übertragen bekommen und somit auch proprietär nutzen können. (Jaeger & Metzger, 2016)

5.3.6 Duale Lizenzierung

Darunter ist zu verstehen, dass Unternehmen ihr Softwareprodukt gleichzeitig unter mehreren Lizenzen distribuieren. Hier kann es sich um das gleichzeitige Verbreiten unter mehreren Open Source Lizenzen handeln um das Kompatibilitätsproblem untereinander zu umgehen. Eine weitere Möglichkeit ist das Softwareprodukt sowohl proprietär als auch frei zur Verfügung zu stellen. Der Vorteil für ein Unternehmen besteht darin, dass dadurch einerseits Umsätze durch Lizenzverkäufe erzielt werden, andererseits durch freie Verbreitung Marktdurchdringung und das Setzen von Standards erreicht werden können. (Jaeger & Metzger, 2016)

Bei der Art der Lizenzierung von eigens entwickelter Software hat der Hersteller freien Handlungsspielraum. Duale Lizenzierung von nicht eigens entwickelter Open Source Software ist allerdings nur dann möglich, wenn es sich nicht um eine Weiterentwicklung eines Softwareprodukts unter einer strengen Copyleft Klausel handelt. Daher ist bei fremden Open Source Lizenzen ohne Copyleft Klausel duale Lizenzierung von Weiterentwicklungen wiederum immer möglich. (Schaaf, 2013)

5.3.7 Unterschiede der OSL

Was die genannten Open Source Lizenzen gemein haben ist, dass sie allesamt dem Kriterienkatalog der Open Source Initiative entsprechen. Unterschiede gibt es jedoch im Umgang mit der Integration in kommerzielle Softwareprodukte, dem Umgang mit Bearbeitungen und möglichen Rechten der vorangegangenen Urheberin bzw. des vorangegangenen Urhebers. (Schaaf, 2013)

Folgende Tabelle nach (Schaaf, 2013) zeigt die Unterschiede auf:

	GPL	LGPL	BSD	Artistic	MPL
Source Code kann uneingeschränkt gelesen, genutzt, bearbeitet und (weiter-) verbreitet werden	x	x	x	x	x
Kann mit proprietärer Software verbunden und ohne Open Source Softwarelizenz (weiter-)verbreitet werden		x	x	x	x
Modifikationen am Open Source lizenzierten Quellcode können im Verbreitungsfall proprietär bleiben			x	x	
Spezielle Rechte für den ursprünglichen Urheber über Bearbeitungen anderer					x

Tabelle 6 - Unterschiede bekannter Open Source Lizenzen

5.3.8 Kompatibilität der OSL untereinander

Bei der Entwicklung von Softwareprodukten ist es nach (Schaaf, 2013) mittlerweile üblich, bestehende Open Source Softwareprodukte in das eigene einzubinden, um Ressourcen zu sparen und fehlendes Wissen zu kompensieren. Dabei ist es möglich, dass verschiedene eingebundene Open Source Softwarekomponenten unter unterschiedlichen Lizenzen stehen. Um dabei rechtliche Verstöße zu vermeiden, müssen Abhängigkeiten und Lizenzbedingungen der einzelnen Komponenten wie folgt geprüft werden:

1. Prüfen der Abhängigkeiten

Bleiben die einzelnen eingebundenen Softwarekomponenten eigenständig und trennbar von allen anderen Komponenten, so besteht keine Gefahr eines Vertragsverstoßes. Eine weitere Prüfung ist daher nicht notwendig. Trifft dies nicht zu, ist auf folgenden Schritt überzugehen. (Schaaf, 2013)

2. Prüfung auf Copyleft Klausel

Maßgeblich für mögliche Vertragsverstöße ist die Copyleft Klausel. Diese bedingt das Veröffentlichen aller Softwarekomponenten unter derselben Copyleft Lizenz, sobald eine Komponente eine solche aufweist. Bleiben die Komponenten von einander jedoch unabhängig wie in Schritt 1 beschrieben, oder keine der eingebundenen Komponenten weist eine Copyleft Klausel auf, so bedarf es keiner weiteren Prüfung. Anderenfalls ist auf den nächsten Schritt überzugehen. (Schaaf, 2013)

3. Prüfung auf Kompatibilität

Bilden die einzelnen Komponenten des Softwareprodukts ein voneinander abgeleitetes Werk und ist unter diesen eine Komponente mit Copyleft Klausel enthalten, so müssen alle Komponenten unter derselben Copyleft Lizenz veröffentlicht werden. Ist dies einer Komponente durch Bedingungen der ursprünglichen Lizenz nicht gestattet, so sind die Lizenzen zueinander nicht kompatibel. Gibt es mehrere Komponenten unter unterschiedlichen Copyleft Lizenzen, so muss auch für diese anhand deren Lizenzbestimmungen festgestellt werden, ob eine derartige Kompatibilitätsklausel enthalten ist. Ist dies wiederum nicht der Fall, sind auch diese Lizenzen miteinander nicht kompatibel. (Schaaf, 2013)

(Fogel, 2013) beschreibt, dass die GPL hierbei als Trennlinie für die erste Kompatibilitätsanalyse herangezogen werden kann, da diese durch ihre Lizenzbedingungen eine Einbindung in proprietären Code definitiv ausschließt.

5.4 Rechtliche Aspekte

Wie in Kapitel 4 erwähnt, gilt für Software das Urheberrechtsgesetz. Einzige Ausnahme stellt Software dar, die so einfach gehalten ist, dass diese nicht den Anspruch der kreativen, geistigen Eigenleistung erfüllt. Für alle anderen Formen von Software gelten durch den urheberrechtlichen Schutz verschiedenste Rechte und Pflichten. (Schaaf, 2013)

Alleinig die Rechteinhaberin bzw. der Rechteinhaber entscheidet darüber, wem er unter welchen Bedingungen Nutzungsrechte an ihrer bzw. seiner Software mittels eines Lizenzvertrages einräumt. Urheberinnen bzw. Urheber von Open Source Softwareprodukten stellen ihre Software mittels einer Open Source Softwarelizenz der Allgemeinheit zur Verfügung, es bedarf daher keines Einzelvertrages zwischen Lizenzgeber und Lizenznehmer. (Schaaf, 2013)

5.4.1 Urheberrecht

Bei Open Source Softwareprodukten ist die Urheberschaft aufgrund der Vielzahl an Softwareentwicklerinnen und Softwareentwicklern oft schwer festzustellen. Wurden Teile der Software unabhängig voneinander entwickelt, so können diese als selbstständige Stücke betrachtet werden, auch hinsichtlich der Urheberschaft. Häufiger ist es jedoch der Fall, dass viele Softwareentwicklerinnen und Softwareentwickler gemeinsam an einem nicht teilbaren Softwareprodukt arbeiten. Daher sind alle beteiligten Softwareentwicklerinnen und Softwareentwickler als Miturheber zu sehen, welche gemeinschaftlich im Konsens über die Verwertung der Software entscheiden. (Schaaf, 2013)

Ist das Open Source Softwareprodukt Ergebnis der geistigen, individuellen Schöpfung der Erstellerin bzw. des Erstellers, so ist diese für sie bzw. ihn urheberrechtlich geschützt. Gleichermaßen trifft das auch auf jede andere Form eines Softwareprodukts zu. Aufgrund der in jeder Open Source Lizenz verankerten „Freiheit“ des Softwareprodukts, gilt die Urheberregelung sowohl für die Erstversion, als auch für jede weitere Bearbeitung. Aufgrund der Vielzahl der entstehenden, teilweise aufeinander aufbauenden Bearbeitungen, empfiehlt es sich in der Praxis hinsichtlich einer möglichen gerichtlichen Verwertbarkeit, die Entstehungskette zu verfolgen. (Bdeiwi, 2016)

5.4.2 Vertragspartner

Ein Hersteller eines Softwareprodukts ist Vertragspartner im Sinne der Lizenzgeberin bzw. des Lizenzgebers, wenn diese bzw. dieser die Rechteinhaberin bzw. der Rechteinhaber des Softwareprodukts ist. Dies trifft bei Open Source Softwarelizenzen nicht zu, da der Hersteller das Softwareprodukt mittels der Open Source Softwarelizenz der Allgemeinheit zur Verfügung stellt. Dadurch hat der Hersteller keine Nutzungsrechte inne, die er der Lizenznehmerin bzw. dem Lizenznehmer weitergeben kann. Somit ist er auch nicht Teil des Lizenzvertrages. (Schaaf, 2013)

5.4.3 Lizenzverstöße

Welche Folgen bei einem Lizenzverstoß eintreten, ist im jeweiligen Lizenzvertrag geregelt. Allerdings trifft das nicht auf alle Open Source Lizenzen zu, wie beispielsweise der BSD Lizenz, welche die Thematik der Folgen von Lizenzverstößen nicht behandelt. Kernthema bei Verstößen gegen Open Source Lizenzbestimmungen ist es, zu beweisen, dass ein vormals freier Source Code nun Bestandteil eines proprietären Softwareprodukts ist. (Schaaf, 2013)

Verwendet eine Lizenznehmerin bzw. ein Lizenznehmer das Open Source Softwareprodukt widersprüchlich der Lizenzbedingungen, kann die Urheberin bzw. der Urheber gegen die Nutzerin bzw. den Nutzer Ansprüche auf Unterlassung, Beseitigung sowie Schadensersatz stellen. Dies trifft auch zu, wenn eine Nutzerin bzw. ein Nutzer gar keinen Lizenzvertrag abgeschlossen hat und die Nutzung außerhalb einer Erschöpfung gemäß des Erschöpfungsgrundsatzes laut Urheberrecht liegt. (Bdeiwi, 2016)

Die Urheberin bzw. der Urheber ist zumeist Urheberin bzw. Urheber der Bearbeitung und nicht der Erstversion. Diese bzw. dieser hat das Risiko, selbst sicherzustellen, dass es sich bei der bearbeiteten Software ursprünglich tatsächlich um ein Open Source Softwareprodukt gehandelt hat. Auch muss die Urheberin bzw. der Urheber der Bearbeitung die zugrundeliegenden Lizenzbedingungen kennen. Stellt sich heraus, dass das ursprüngliche Softwareprodukt kein Open Source Softwareprodukt ist, ist ein Benutzen und Bearbeiten nicht ohne dementsprechende Lizenz erlaubt. Diese muss bei der Urheberin bzw. beim Urheber oder der entsprechenden Rechteinhaberin bzw. dem entsprechenden Rechteinhaber bezogen werden. Handelt sich bei der Software um ein Open Source Softwareprodukt, ist die Urheberin bzw. der Urheber der Bearbeitung gleichermaßen schadenersatzpflichtig wie eine Nutzerin bzw. ein Nutzer, wenn diese bzw. dieser keine Lizenz erworben oder sich widersprüchlich der Lizenzbestimmungen verhalten hat. Laut deutschem und österreichischem Recht können Haftungs- sowie Gewährleistungsansprüche nicht vollständig ausgeschlossen werden, daher bergen auch Verpflichtungen laut Lizenzvertrag gegenüber der Nutzerin bzw. dem Nutzer ein Risikopotential. (Bdeiwi, 2016)

5.4.4 Gültigkeit

Die Gültigkeit von Open Source Lizenzen und deren vertraglichen Bedingungen wurde bereits durch amerikanische als auch europäische Gerichte bestätigt und sind daher rechtlich bindend. Besonders zu erwähnen sind hier Softwareprodukte unter Open Source Lizenzen, die von einer Dienstnehmerin bzw. einem Dienstnehmer im Auftrag des Dienstgebers umgesetzt wurden. In diesem Fall hat der Dienstgeber die Nutzungsrechte an dem erstellten Softwareprodukt inne. Verstößt dieser jedoch gegen lediglich einen Punkt der Open Source Lizenzbedingungen, verliert dieser sämtliche Nutzungsrechte an die Dienstnehmerin bzw. den Dienstnehmer, welche Urheberin bzw. welcher Urheber des Softwareprodukts ist. Dieser Umstand ist auf Gerichtsentscheidungen deutscher Gerichte hinsichtlich Urteile zur GPL zurückzuführen. (Schaaf, 2013)

5.5 Vorteile und Nachteile von Open Source Software

Durch den Einsatz von Open Source Softwareprodukten verzichtet ein Unternehmen zwar auf eine speziell angepasste Softwarelösung, erhält dafür aber ein Softwareprodukt, welches unter Umständen bewährt, weit verbreitet und bereits verbessert oder weiterentwickelt ist. Außerdem stellt die Community, die durch das Open Source Softwareprodukt angesprochen und einbezogen werden kann, einen positiven Faktor dar. Auch der Kostenfaktor kann bei der

Entscheidung zu einem Open Source Softwareprodukt eine maßgebliche Rolle spielen. (Bdeiwi, 2016)

Ebenso bietet die Teilnahme an Open Source Softwareprojekten Softwareentwicklerinnen und Softwareentwicklern die Möglichkeit, soziale Bedürfnisse zu decken.

Fasst man (Cruz, 2012), (Hennig, 2009), (Schaaf, 2013), (Storz, 2012) und (Weber, 2005) zusammen, ergibt sich folgende Übersicht von Vor- und Nachteilen hinsichtlich Open Source Software:

Vorteile	**Nachteile**
Anpassbarkeit	Betriebskosten
Erweiterbarkeit	Produktsupport
Kosten	Entwicklerzentrierung
Unabhängigkeit	Schulungsaufwand
Offene Standards	Weiterentwicklung
Sicherheit	Anwendungssoftware
Qualität und Stabilität	Interoperabilität mit kommerzieller Software
Support durch die Community	Rechtliche Risiken

Tabelle 7 - Vorteile und Nachteile von Open Source Software

5.5.1 Vorteile von Open Source Software

Die gelisteten Vorteile von Open Source Softwareprodukten werden wie folgt näher erläutert.

Anpassbarkeit - Anpassbarkeit an die Geschäftsprozesse

Ein Problem, das proprietäre Softwareprodukte oft mit sich bringen, ist die eingeschränkte Anpassbarkeit der Prozesse. Damit ist gemeint, dass man sich bei proprietärer Software für eine Variante oder eine der vorgegebenen Varianten der Prozessabbildung entscheiden muss, und dementsprechend die eigenen Geschäftsprozesse anpassen muss. Open Source Softwareprodukte hingegen können durch ihr Wesen bestmöglich an die eigenen Geschäftsprozesse angepasst werden. Diese Anpassungen gehen mit Entwicklungskosten einher. Betrachtet man jedoch die versteckten Kosten, die durch Geschäftsprozessanpassungen von Unternehmen mit großer Mitarbeiterzahl anfallen, bietet das Open Source Softwareprodukt wiederum einen langfristigen Kostenvorteil. (Storz, 2012)

Erweiterbarkeit der Funktionalität

Während man hinsichtlich der Erweiterung der Produktfunktionalität bei proprietären Softwareprodukten an den Hersteller gebunden ist, bieten Open Source Softwareprodukte hier mehr Freiheiten hinsichtlich der Möglichkeit zur eigenen Weiterbearbeitung oder Zusammenarbeit mit der Community (Storz, 2012). (Cruz, 2012) beschreibt dies als optimale Flexibilität.

Kosten - Total Cost of Ownership

Das Nichtanfallen von Lizenzgebühren bedeutet nicht, dass die Verwendung eines Open Source Softwareprodukts kostenlos ist, denn die monetären Aufwände für Inbetriebnahme, Wartung und weitere Tätigkeiten rund um das Open Source Softwareprodukt dürfen nicht außer Acht gelassen werden. Dass für Open Source Softwarelizenzen keine direkten Lizenzgebühren erhoben werden dürfen, ist daher ein Vorteil (Cruz, 2012). Dennoch ist mit einem durchschnittlichen Gesamtkostenvorteil gegenüber proprietärer Software von ca. 50% zu rechnen. Selbst wenn der Kostenfaktor gegenüber einem proprietären Softwareprodukt zum Zeitpunkt der Inbetriebnahme nur unwesentlich abweicht, ist der Kostenvorteil über einen längeren Zeitraum betrachtet aufgrund des offenen Source Codes und der damit verbundenen Erweiterbarkeit nicht von der Hand zu weisen. (Hennig, 2009) (Storz, 2012)

Unabhängigkeit - Vendor Look

Hat man sich erstmal für ein proprietäres Softwareprodukt entschieden, ist man an den Anbieter gebunden. Auf das Softwareprodukt an sich bezogen hat man beispielsweise wenig Mitsprachemöglichkeit auf technische oder funktionale Entwicklungen. Aber auch wenn man mit dem Softwareprodukt zufrieden ist, ist man beim Dienstleistungsangebot rund um das Softwareprodukt eingeschränkt. (Storz, 2012)

Offene Standards

Durch den freien Zugang zum Source Code und den damit verbundenen offenen Standards, kann man die Vorteile der Open Source Community nutzen und von der ausgelagerten Weiterentwicklung des Open Source Softwareprodukts profitieren. (Cruz, 2012)

Sicherheit - Investitionssicherheit

Die hier zu stellende Grundfrage lautet: Was passiert, wenn der Anbieter meines Softwareprodukts nicht mehr existiert? Diese Frage ist sowohl hinsichtlich proprietärer Software als auch Open Source Softwareprodukten zu stellen. Verschwindet der proprietäre Anbieter, gibt es möglicherweise niemanden mehr, der das Softwareprodukt weiterentwickelt bzw. Serviceleistungen anbietet. Dies gilt aber ebenso für Open Source Softwareprodukte mit kleiner Community, für die es durch verschwinden der Community möglicherweise keine externen Entwicklungen mehr geben wird. (Storz, 2012)

Qualität und Stabilität

Durch den frei zugänglichen Source Code ist es jedem möglich, diesen zu untersuchen. Somit können durch jede Softwareentwicklerin und jeden Softwareentwickler Fehler gefunden und behoben werden. Dies hat sich bei großen Open Source Softwareprojekten durch die Vielzahl der Beteiligten an der Community, als ein Hauptfaktor für Qualität und Stabilität herausgestellt. Wichtig ist jedoch nicht alleinig die Größe der Community, sondern vielmehr die Verbreitung des Know-Hows. (Brügge, 2012) (Cruz, 2012)

Support durch die Community

Hinter erfolgreichen Open Source Softwareprojekten steht die Community. Ist ein Open Source Softwareprodukt weit am Markt verbreitet, so kann man davon ausgehen, dass auch die

Community hinter dem Produkt entsprechend groß ist. Dadurch wächst auch die Know-How Verbreitung innerhalb der Gruppe der Softwareentwicklerinnen und Softwareentwicklern, was wiederum auf schnelle Reaktionen bei Supportanfragen schließen lässt. (Weber, 2005)

5.5.2 Nachteile von Open Source Software

Die gelisteten Nachteile von Open Source Softwareprodukten werden wie folgt näher erläutert.

Betriebskosten

Obwohl für Open Source Softwareprodukte keine direkten Lizenzgebühren zu entrichten sind, muss sich ein Unternehmen dennoch mit Kosten der Installation, Wartung sowie möglicher Migration und eigenständiger Weiterentwicklung auseinandersetzen. Ob sich der Einsatz eines Open Source Softwareprodukts aus Kostensicht gegenüber der Entrichtung von Lizenzgebühren für proprietäre Softwareprodukte lohnt, hängt daher auch mit dem vorhandenen Know-How des eigenen Unternehmens zusammen. (Bridge, 2013) (Storz, 2012)

Produktsupport

Der Vorteil kommerzieller, proprietärer Softwareprodukte, entsprechenden Support durch den Hersteller erhalten zu können, geht bei der Entscheidung zu einem Open Source Softwareprodukt verloren. Zwar erhält man Hilfestellung durch die Community über Foren im Internet, dies entspricht jedoch nicht den Möglichkeiten des Supports durch den Hersteller. Das Unternehmen ist somit selbst für das Open Source Softwareprodukt und dessen Installation, Wartung und benötigte Weiterentwicklung verantwortlich. (Ecomsolutions, 2007)

Entwicklerzentrierung

Obwohl die Größe der Community, und damit die Anzahl an beteiligten Softwareentwicklerinnen und Softwareentwicklern, mit der Größe und Bekanntheit des entsprechenden Open Source Softwareprojektes steigt, gibt es dennoch ein Kernteam an Beteiligten, die einerseits Know-How über das Gesamtprojekt bündeln, und deren Motivation ausschlaggebend für die weitere Entwicklung des Open Source Softwareprojektes ist (Weber, 2005). (Bridge, 2013) fügt weiter hinzu, dass sich die Interessen und Vorstellungen von Nutzerinnen und Nutzer hinsichtlich der Weiterentwicklung des Open Source Softwareprodukts stark von denen der Softwareentwicklerinnen und Softwareentwicklern unterscheiden können.

Schulungsaufwand

Entscheidet man sich für den Einsatz eines Open Source Softwareprodukts, kann es zu Engpässen hinsichtlich der Findung qualifizierten Personals kommen. Dieses lässt sich hauptsächlich über die Community eruieren, da man am regulären Markt leichter Softwareentwicklerinnen und Softwareentwickler mit Know-How über proprietärer, kommerzieller Softwareprodukte findet. Dementsprechend ist die Suche nach qualifiziertem Schulungspersonal auch von der Verbreitung des Open Source Softwareprodukts und der damit verbundenen Größe der Community abhängig. (Schaaf, 2013)

Weiterentwicklung

Bei Open Source Softwareprodukten ist es schwieriger abzuschätzen, in welche Richtung und mit welcher Geschwindigkeit sich dieses weiterentwickelt, als bei kommerziellen Softwareprodukten. Da sich Softwareentwicklerinnen und Softwareentwickler meist aus sozialen Gründen für die Teilnahme an einem Open Source Softwareprojekt entscheiden, werden diese nur solange an dem Projekt mitarbeiten, solange sie ihre sozialen Bedürfnisse mittels des entsprechenden Projektes abdecken können (Weber, 2005). Schwindet die Motivation des Kerns der Community, kann dies bis zur Stagnation der Weiterentwicklung des Open Source Softwareprodukts durch die Community führen (Ecomsolutions, 2007). Ist dies der Fall, ist das Unternehmen entweder auf Mitarbeiter im eigenen Unternehmen mit entsprechendem Know-How angewiesen, oder auf externe Leistungen. (Schaaf, 2013)

Anwendungssoftware

Die Suche nach einem geeigneten Open Source Softwareprodukt für den Einsatz im eignen Unternehmen gestalten sich schwieriger als die Suche nach passenden proprietären Softwareprodukten. Dies liegt einerseits am schwer überschaubaren Markt von Open Source Softwareprodukten, andererseits kann diese für potentielle Anwenderinnen und Anwender eine Herausforderung darstellen, da Open Source Softwareprodukte in der Regel schlechter dokumentiert sind als proprietäre, kommerzielle Softwareprodukte. Zusätzlich wird der Einsatz von Open Source Softwareprodukten dadurch erschwert, dass diese Seiteneffekte auf andere Softwareprodukte haben können, welche vom Unternehmen eingesetzt werden. (Schaaf, 2013)

Interoperabilität mit kommerzieller Software

Hersteller proprietärer, kommerzieller Softwareprodukte haben oft kein Interesse daran, Schnittstellen und Dateiformate ihres Softwareprodukts offenzulegen. Dadurch ist es Softwareentwicklerinnen und Softwareentwicklern von Open Source Softwareprojekten nicht möglich, Kompatibilität zu proprietären, kommerziellen Softwareprodukten zu schaffen. Ob dies eher als Nachteil für Open Source Softwareprodukte oder für kommerzielle, proprietäre Softwareprodukte zu sehen ist, hängt auch mit der Marktsituation zusammen. (Schaaf, 2013)

Rechtliche Risiken

Die dezentralisierte Entwicklungsweise von Open Source Softwareprojekten durch die Community birgt auch rechtliche Risiken für deren Nutzerinnen und Nutzer. Wurde in ein Open Source Softwareprodukt unerlaubter Weise Source Code eines proprietären Softwareprodukts verwendet, könnte die Rechteinhaberin bzw. der Rechteinhaber des proprietären Softwareprodukts Schadenersatzforderungen geltend machen. (Bdeiwi, 2016) (Schaaf, 2013)

Der wohl wichtigste rechtliche Aspekt stellt die Copyleft Klausel dar. Enthält ein proprietäres Softwareprodukt Elemente eines Open Source Softwareprodukts, welches unter einer Open Source Softwarelizenz mit strenger Copyleft Klausel lizenziert wurde, muss das gesamte bislang proprietäre Softwareprodukt unter derselben Open Source Softwarelizenz mit strengem Copyleft freigegeben werden, und somit auf der Source Code des bislang proprietären Softwareprodukts. Des Weiteren ist auch hier mit Schadenersatzansprüchen zu rechnen. Das Geschäftsmodell des Herstellers der proprietären Software könnte dadurch stark gefährdet werden. Demzufolge stellen

unübersichtliche Bearbeitungsketten für ein Unternehmen, das ein Open Source Softwareprodukt nutzen und weiterbearbeiten möchte, das größte Risikopotential dar, da es selbst für die rechtmäßige Nutzung des Softwareprodukts verantwortlich ist. Gelingt es dem Unternehmen jedoch, alle Bearbeitungen eines Open Source Softwareprodukts zu verfolgen, Lizenzbedingungen genau zu prüfen und sich an diese zu halten, stellt die Nutzung und die weitere Bearbeitung kein spezifisches rechtliches Risiko dar. (Bdeiwi, 2016) (Schaaf, 2013)

5.6 Fazit

Das Wesen eines Open Source Softwareprodukts ist der freie Source Code. Mit frei ist dabei gemeint, dass dieser Source Code frei zugänglich ist, sowie weiterbearbeitet und weiterverbreitet werden darf. Die Free Software Foundation sowie die Open Source Initiative bilden dabei Institutionen, welche die Bedingungen freier Software genauer definieren, Open Source Softwarelizenzen herausgeben und sich für die Einhaltung der darin enthaltenen Bedingungen einsetzen. Das Wesen solcher Softwarelizenzen, die nicht-kommerzielle Einstellung, die gemeinschaftliche Entwicklungsleistung sowie das dezentralisierte Arbeiten fassen die Besonderheiten von Open Source Software zusammen. Trotz der nicht-kommerziellen Einstellung ist es dennoch möglich Open Source Software kostenpflichtig zu vertreiben, lediglich für die Lizenzen selbst dürfen keine Gebühren verlangt werden.

Neben dem freien Source Code stellt die Community der Entwicklerinnen und Entwickler das Kernelement der Open Source Softwareentwicklung dar. Diese nehmen auf unterschiedliche Art und Weise am Entwicklungsprozess teil, um über geographische und kulturelle Grenzen hinweg gemeinschaftlich zu arbeiten. Dabei stehen soziale Faktoren wie der Wunsch nach Anerkennung und Selbstverwirklichung im Vordergrund.

Mit der Erfolgsgeschichte von Linux in den neunziger Jahren begann Open Source Software für den wirtschaftlichen und öffentlichen Bereich relevant zu werden. Eine wichtige Rolle spielte dabei Richard Stallman, der mit der Gründung des GNU Projektes 1984 und der Gründung der Free Software Foundation 1985 den Grundstein für viele weitere Projekte und die Entstehung von Open Source Softwarelizenzen legte. Ein weiterer Protagonist war Linus Torvalds, der als Gründer des Linux Projektes gilt. Einen Entwicklungsschritt stellte die Definition des Begriffes „Open Source“ dar, welcher durch die Protagonisten der 1998 gegründeten Open Source Initiative deklariert wurde, um Klarheit und Abgrenzung zum Begriff „frei“ im Sinne von kostenlos zu schaffen. Darauf aufbauend entwickelten sich bis heute viele verschiedene Open Source Softwareprojekte, und verschiedene Arten von Open Source Softwarelizenzen.

Zu den verschiedenen Open Source Softwarelizenzen gehören Lizenzen mit strengem Copyleft, beschränktem Copyleft, ohne Copyleft, Wahlmöglichkeiten, Sonderrechten sowie Lizenzen mit der Möglichkeit dualer Lizenzierung. Der Begriff des Copylefts beschreibt dabei die Lizenzänderungsmöglichkeit bei Bearbeitung und weiterer Distribution eines Open Source Softwareprodukts. Diese fällt je nach Art der Open Source Softwarelizenz unterschiedlich aus. Ein strenges Copyleft hat zur Folge, dass jegliche Weiterverbreitung einer Bearbeitung eines Open Source Softwareprodukts mit denselben Lizenzbedingungen zu erfolgen hat, wie das

ursprüngliche Open Source Softwareprodukt. Lizenzen mit beschränktem Copyleft ist es möglich die ursprüngliche freie Software mit Bearbeitungen unter anderen Lizenzbedingungen zu kombinieren. Lizenzen ohne Copyleft Klausel erlauben jegliche Weiterentwicklung in unfreie Eigenentwicklungen einzubinden, sowie eine Bearbeitung eines Open Source Softwareprodukts, welche unter einer Lizenz ohne Copyleft Klausel steht, proprietär weiterverbreiten. Bei Lizenzen mit Sonderrechten bekommt ein Unternehmen die Rechte an Weiterentwicklungen durch die Community übertragen und darf diese somit auch proprietär nutzen können. Bei dualer Lizenzierung distribuiert ein Unternehmen sein Softwareprodukt gleichzeitig unter mehreren Lizenzen. Hier kann es sich um das gleichzeitige Verbreiten unter mehreren Open Source Lizenzen handeln oder es sowohl proprietär als auch frei zur Verfügung gestellt werden. In jedem Fall ist bei Einsatz von mehreren Open Source Lizenzen deren Kompatibilität untereinander zu prüfen. Urheberinnen bzw. Urheber von Open Source Softwareprodukten stellen ihre Software mittels einer Open Source Softwarelizenz der Allgemeinheit zur Verfügung, es bedarf daher keines Einzelvertrages zwischen Lizenzgeber und Lizenznehmer.

Die GPL Lizenz, welche zu den Lizenzen mit strengem Copyleft gehört, war von Beginn des Bestehens von Open Source Softwarelizenzen an hinsichtlich der Häufigkeit ihrer Verwendung an erster Stelle. Seit 2016 wurde diese aber von der MIT Lizenz, welche zu den Open Source Softwarelizenzen ohne Copyleft Klausel gehört, auf Platz zwei verdrängt. Dies lässt darauf schließen, dass immer mehr Nutzerinnen und Nutzer von Open Source Softwareprodukten diese proprietär nutzen möchten, ohne dabei rechtliche Risiken einzugehen.

Der Einsatz von Open Source Softwareprodukten bietet Vorteile aber auch Nachteile. Bei den Vorteilen handelt es ich um Anpassbarkeit, Erweiterbarkeit, Kostenersparnissen, Unabhängigkeit, Offene Standards, Sicherheit, Qualität, Stabilität, sowie Support durch die Community. Dem gegenüber stehen die Nachteile, welche in den Bereichen der Betriebskosten, Produktsupport, Entwicklungszentrierung, Schulungsaufwand, Weiterentwicklung, Anwendungssoftware, Interoperabilität und rechtlichen Risiken wiederfinden, wobei die rechtlichen Risiken Hinsichtlich der Forschungsfrage den relevantesten Faktor darstellen. Kernthema bei Verstößen gegen Open Source Lizenzbestimmungen ist es, zu beweisen, dass ein vormals freier Source Code nun Bestandteil eines proprietären Softwareprodukts ist. Die Copyleft Klausel stellt hierbei das größte Risiko für Unternehmen dar. Setzt dieses nämlich Open Source Softwareprodukte mit strengem Copyleft in ihrem proprietären Softwareprodukt ein, ist mit Unterlassungsansprüchen und Schadenersatzleistungen zu rechnen. Um dies zu vermeiden muss das ehemals proprietäre Softwareprodukt unter derselben Lizenz wie das Open Source Softwareprodukt freigegeben werden, wodurch dessen Source Code ebenso frei zugänglich zu machen ist. Dies kann das Geschäftsmodell des Herstellers der betreffenden Software gefährden.

6 OPEN SOURCE SOFTWARE IN UNTERNEHMEN

Obwohl für Open Source Softwareprodukte keine Lizenzgebühren eingehoben werden dürfen, fand und findet das Modell der Open Source Software dennoch gezielte Unterstützung durch Unternehmen. Ein bekannter Vertreter hierfür ist der Konzern IBM, der beispielsweise durch sein Engagement für Linux dem Konkurrenten Microsoft Marktanteile abringen konnte. Gleichermaßen wurden von diesem Konzern das Apache Projekt unterstützt, und einige vormals proprietäre Softwareprodukte wie beispielsweise die Entwicklungsumgebung „Eclipse“ als Open Source Softwareprodukte freigegeben. Jedoch beteiligen sich nicht nur große Konzerne, sondern auch viele kleinere Unternehmen am Open Source Modell. (Brügge, 2012)

Auch bei der Entwicklung proprietärer Softwareprodukte durch Unternehmen, können die Vorteile von Open Source Softwareprodukten genutzt werden. Dabei ist es essentiell, auf die Kompatibilität der Softwarelizenzen untereinander zu achten. Auch die Beteiligung an der Open Source Communitiy ist ein wichtiger Faktor, den Unternehmen dabei nicht außer Acht lassen dürfen. (Brügge, 2012)

Hinsichtlich der Art der Beteiligung können laut (Brügge, 2012) folgende Szenarien unterschieden werden:

- Weiterentwicklung von existierender Open Source Software
- Freigabe von ehemaliger proprietärer Software als Open Source Software
- Veranlassung eines neuen Open Source Projektes

Wichtiger als die Unterscheidung der Szenarien ist laut (Brügge, 2012) jedoch die Positionierung des Unternehmens selbst.

6.1 Positionierung des Unternehmens zu OSS

Die Positionierung des Unternehmens kann laut (Brügge, 2012) in folgenden vier Formen auftreten:

- Das Unternehmen stellt Open Source Software als Hauptgeschäftszweig her
- Das Unternehmen stellt Komplemente zur Open Source Software her
- Das Unternehmen nutzt Open Source Software für seine internen Prozesse
- Das Unternehmen nutzt Open Source Software in proprietären Produkten

Unabhängig der Positionierung des Unternehmens ist die Möglichkeit zur kollaborativen Nutzung, Bearbeitung und Weiterverbreitung der Open Source Software durch die Community der Hauptaspekt für die Entwicklung bzw. den Einsatz von Open Source Softwareprodukten durch Unternehmen. Die Mitgliederzahl der Community und deren Dichte des technischen Know-Hows

ist dabei die entscheidende Größe für die Erzielung der mit Open Source Software verbundenen Vorteile. Dennoch überwiegen die Vorteile von Open Source Software nicht immer den mit ihr verbundenen Nachteilen. Hierbei spielt die Positionierung des Unternehmens zur betreffenden Open Source Software eine wichtige Rolle in der Entscheidung, ob Software als Open Source Software zur Verfügung gestellt wird oder für welche Open Source Lizenz sich das Unternehmen entscheidet. (Brügge, 2012)

6.1.1 Entwicklung von OSS als Hauptgeschäftszweig

Da einige Open Source Softwareprodukte mit Funktionalität und Qualität proprietärer Softwareprodukte mithalten können, wird Open Source Software als Geschäftsmodell von bekannten Anbietern wie beispielsweise Red Hat, IBM und Oracle realisiert. Verdient wird im Gegensatz zu proprietärer Software nicht direkt an den Lizenzverkäufen, sondern die Kommerzialisierung des Open Source Softwareprodukts erfolgt durch Serviceleistungen rund um das Open Source Softwareprodukt oder durch das Anbieten zusätzlicher Module. Aus Kundensicht ist nebst der Kostenersparnis vor allem die Unabhängigkeit und Flexibilität interessant, die durch den freien Source Code des Open Source Softwareprodukts erreicht wird. (Storz, 2012)

Des Weiteren sind nach (Brügge, 2012) folgende Szenarien möglich, die Softwarehersteller dazu bringen, ihre Entwicklungsleistung unter Open Source Softwarelizenzen zu stellen:

- Auftragsentwicklung,
- Duale Lizenzierung, sowie
- Freigabe bei Supportaufgabe

Bei Auftragsentwicklung kommt der Wunsch nach Freigabe unter einer Open Source Softwarelizenz entweder vom Kunden, oder ist dadurch bedingt, dass Weiterentwicklungen an Open Source Softwareprodukten vorgenommen werden sollen, welche sich unter einer Open Source Softwarelizenz mit strenger oder beschränkter Copyleft Klausel befinden. (Brügge, 2012)

Bei Dualer Lizenzierung können wie in Kapitel 6.3.6 beschrieben sowohl die Vorteile proprietärer Softwareprodukte als auch die der Open Source Softwareprodukte genutzt werden. Dadurch können Bezieher des Softwareprodukts unter einer entsprechenden Open Source Softwarelizenz dazu verpflichtet werden, sämtliche Weiterentwicklungen ebenso frei zur Verfügung zu stellen. Möchte ein Unternehmen dies vermeiden, kann es eine proprietäre Lizenz beziehen. Dadurch können einerseits Standards und Marktdurchdringung und andererseits Gewinne durch Lizenzverkäufe generiert werden. (Jaeger & Metzger, 2016)

Bei proprietären Softwareprodukten trägt die Kundin bzw. der Kunde mitunter das Risiko, dass dieses durch Verschwinden des Herstellers vom Markt oder dessen Beendigung von Supportleistungen nicht mehr gewartet und weiterentwickelt wird. Um potentiellen Kundinnen und Kunden Sicherheit zu geben und positive Reputation zu erlangen, vereinbaren Hersteller oftmals im Vorhinein, das betreffende proprietäre Produkt nach einigen Jahren frei zur Verfügung zu stellen. (Brügge, 2012)

6.1.2 Entwicklung von Komplementen zu OSS

Hinter diesem Geschäftsmodell steht die Idee, an der Weiterentwicklung eines Open Source Softwareprodukts beizutragen, um dadurch Komplemente zum betreffenden Softwareprodukt vertreiben zu können. Durch den Vertrieb der Komplemente sollen die Einnahmen und Gewinne des Unternehmens erzielt werden. Dabei muss durch den Beitrag an der Weiterentwicklung des Softwareprodukts sichergestellt werden, dass das Komplement so eng mit der Open Source Software in Verbindung gebracht wird, dass diese zusammen als Einheit als mehr wert wahrgenommen werden, als die einzelnen Stücke selbst. Dementsprechend hegt das Unternehmen Interesse daran, ein qualitativ hochwertiges und stabiles Open Source Softwareprodukt weiterzuentwickeln, da bei dessen Verbreitung ebenso die Nachfrage des Komplementes steigt. Beispiele für solche Komplemente sind etwa proprietäre Softwareprodukte, Hardware oder Supportleistungen. (Brügge, 2012) (Haruvy, Sethi, & Zhou, 2008)

6.1.3 Einsatz von OSS für interne Prozesse

Software wird nicht nur für den Markt entwickelt, sondern wird innerhalb von Unternehmen ebenso für die Abdeckung der eigenen Bedürfnisse entwickelt. Gibt man diese Entwicklung als Open Source Softwareprodukt frei, können Unternehmen dabei von den in Kapitel 6.5.1 genannten Vorteilen profitieren, vor allem durch offene Standards und den Support durch die Community. Auch kann sich die Qualität und Sicherheit des Softwareprodukts verbessern, da Softwareentwicklerinnen und Softwareentwickler Interesse daran hegen könnten, diese zu verbessern. Ob die Freigabe von internen Entwicklungen als Open Source Softwareprodukt sinnvoll ist, hängt jedoch stark von deren Funktion und der Marktsituation ab. Ist dabei etwa mit dem Verlust von Wettbewerbsvorteilen zu rechnen, da über das Softwareprodukt Rückschlüsse auf wichtige Geschäftsprozesse des Unternehmens gezogen werden können, ist von der Offenlegung des Source Codes abzuraten. (Brügge, 2012)

Ebenso können Unternehmen von der Art und Weise wie Open Source Software entsteht und weiterentwickelt wird, profitieren, ohne dabei tatsächlich ein internes Softwareprodukt frei zur Verfügung zu stellen. Dies soll erzielt werden, indem das interne Softwareprodukt innerhalb des Unternehmens gleich eines Open Source Softwareprodukts quelloffen zur Verfügung gestellt und behandelt wird, und jede Mitarbeiterin und jeder Mitarbeiter die Möglichkeit hat, zu dessen Weiterentwicklung beizutragen. (Smith, 2016) (Vaughan-Nichols, 2013)

6.1.4 Nutzung von OSS in proprietären Produkten

Bei der Entwicklung von proprietärer Software durch Unternehmen kann der Bedarf an Softwarekomponenten entstehen, welche bereits mittels eines Open Source Softwareprojektes umgesetzt wurden. Diese Komponenten können in das eigene proprietäre Softwareprodukt eingebunden werden, um Zeit und Kosten für die entsprechende initiale Eigenentwicklung der Komponenten einzusparen. Dabei muss das Open Source Softwareprodukt besonders

hinsichtlich dessen Lizenz und der damit verbundenen Kompatibilität mit dem eigenen proprietären Softwareprodukt geprüft werden. (Brügge, 2012)

Um Open Source Softwareprodukte im eigenen proprietären Softwareprodukt zu nutzen und sich an der Community des entsprechenden Open Source Softwareprodukts als Unternehmen zu beteiligen, beschreibt (Brügge, 2012) folgende Vorgehensweise:

- Genaue Prüfung und Spezifikation der Anforderungen an das potentielle Open Source Softwareprodukt
- Analyse potentieller Open Source Softwareprodukte entsprechend der Anforderungen und Lizenzbedingungen
- Entscheidung für das Einbinden der Komponente
- Entwicklung einer neuen Komponente als Verbindungsstück zwischen der eigenen Software und dem Open Source Softwareprodukt
- Anpassen des Open Source Softwareprodukts an die eigenen Bedürfnisse
- Übermittlung der Anpassungen des Open Source Softwareprodukts an die Community zur Begutachtung
- Laufende Synchronisation der eigenen Anpassungen und der Anpassungen durch die Community

In der Phase der Anpassung des Open Source Softwareprodukts ist besonders auf Änderungen zu achten, welche allgemeine Gültigkeit besitzen. Dabei handelt es sich zumeist um die Behebung von Fehlern, welche an die Community zurückfließen sollten. (Brügge, 2012)

Die laufende Synchronisation der Anpassungen spiegelt die aktive Beteiligung an der Community wieder (Brügge, 2012).

Die Alternative dazu ist das sogenannte Forking. Dabei handelt es sich allgemein um die Aufspaltung eines ursprünglichen Softwareprojektes in ein oder mehrere neue Softwareprojekte. Dies tritt bei Open Source Softwareprojekten beispielsweise ein, wenn sich die Communitiy uneinig über die zukünftige Ausrichtung des Open Source Softwareprojektes ist und sich schließlich spaltet. In Hinblick auf den Einsatz von Open Source Software in proprietärer Software könnte dies auftreten, wenn das Unternehmen das bestehende Open Source Softwareprodukt ohne Einbindung der Community selbst weiterentwickelt und auf die eigenen Bedürfnisse angepasst wird. (Rouse, whatis.techtarget.com, 2014)

Sowohl technische als auch wirtschaftliche Aspekte sprechen für den Einsatz von Open Source Softwarekomponenten. Durch ihren Einsatz können Unternehmen Kosten und Zeit einsparen, da sie die gewünschte Komponente nicht selbst entwickeln lassen müssen. Des Weiteren trägt der Einsatz von Open Source Softwarekomponenten zur Bildung von Standards bei. Dies wiederum bringt eine Minimierung des Risikos mit sich, da eine weit verbreitete Open Source Softwarekomponente durch ihren vielfachen Einsatz bereits getestet, verbessert und inhaltlich bekannt ist. Für die Softwareentwicklerinnen und Softwareentwickler bedeutet dies, dass die allgemeine Qualität der Open Source Softwarekomponente hinsichtlich ihrer Leistung und

Stabilität bekannt ist, und ein Erlernen des Umgangs aufgrund der Standardisierung vereinfacht wird. (Brügge, 2012)

6.1.5 Vorteile und Nachteile von OSS nach Positionierung des Unternehmens

Abhängig von der Positionierung des Unternehmens, ergeben sich neben den in Kapitel 6.5.1 erläuterten Vor- und Nachteilen von Open Source Softwareprodukten, folgende zusätzlichen Vor- und Nachteile:

Positionierung des Unternehmens	**Mögliche Vorteile**	**Mögliche Nachteile**
Entwicklung von OSS als Hauptgeschäftszweig	Querfinanzierung über Komplemente möglich Steigerung des Marktanteils sowie Finanzierung durch Duale Lizenzierung Positive Reputation und Kundengewinnung durch Vereinbarte OS Freigabe bei Beendigung des Supports	Finanzierung abhängig von erfolgreicher Querfinanzierung Bindung an Lizenztyp bei Weiterentwicklung
Entwicklung von Komplementen zu OSS	Steigerung der Verkaufszahlen des Komplementes durch Steigerung des Marktanteils der OSS	Investitionen in die Weiterentwicklung der OSS bewirken nicht die dadurch gewünschten gesteigerten Verkaufszahlen des Komplementes
Einsatz von OSS für interne Prozesse	Behebung von Sicherheitslücken durch die Community Intensivierung von Kundenbeziehungen	Verlust von Wettbewerbsvorteilen Offenlegung von Sicherheitslücken
Nutzung von OSS in proprietären Produkten	Ressourcenersparnis	Einbindung von OSS mit strengem Copyleft nicht möglich Einbindung von OSS mit beschränktem Copyleft problematisch Eventuelle Lizenzverletzungen in ausgewählter OSS vorhanden Forking

Tabelle 8 - Vor- und Nachteile von OSS bezogen auf die Positionierung des Unternehmens

Hinsichtlich der Forschungsfrage widmen wir uns folglich der Positionierungsvariante, bei welcher Unternehmen Open Source Softwareprodukte in ihren eigenen proprietären Softwareprodukten verwenden.

6.2 Präzedenzfälle

Um die rechtmäßige Verwendung von Open Source Softwarelizenzen durchzusetzen, wurde dieses Thema bereits mehrfach unter anderem an europäischen und US-amerikanischen Gerichten verhandelt.

Aufgrund der Erläuterungen der vorangegangenen Kapitel ergibt sich, dass Open Source Softwareprodukte mit strengem Copyleft nicht in proprietäre Softwareprodukte eingebunden werden dürfen, wenn diese proprietär bleiben sollen. Auch die Einbindung von Open Source Softwareprodukten mit beschränktem Copyleft bringt Auflagen mit sich, die eingehalten werden müssen, um die Lizenzbedingungen nicht zu verletzen. Dementsprechend beschäftigen sich Gerichte und Interessensvertreter mit diesen beiden Open Source Softwarelizenztypen und Lizenzverletzungen, die mit der Einbindung von Open Source Softwareprodukten unter diesen Lizenzen in proprietäre Softwareprodukte in Zusammenhang stehen.

Folgender Auszug an Präzedenzfällen gibt einen Überblick zu Verhandlungen bzgl. der rechtmäßigen bzw. unrechtmäßigen Verwendung von Softwareprodukten unter der GPL und LGPL Lizenz, welche laut Abbildung 7 die am häufigsten verwendeten Vertreter der betreffenden Open Source Softwarelizenzen sind.

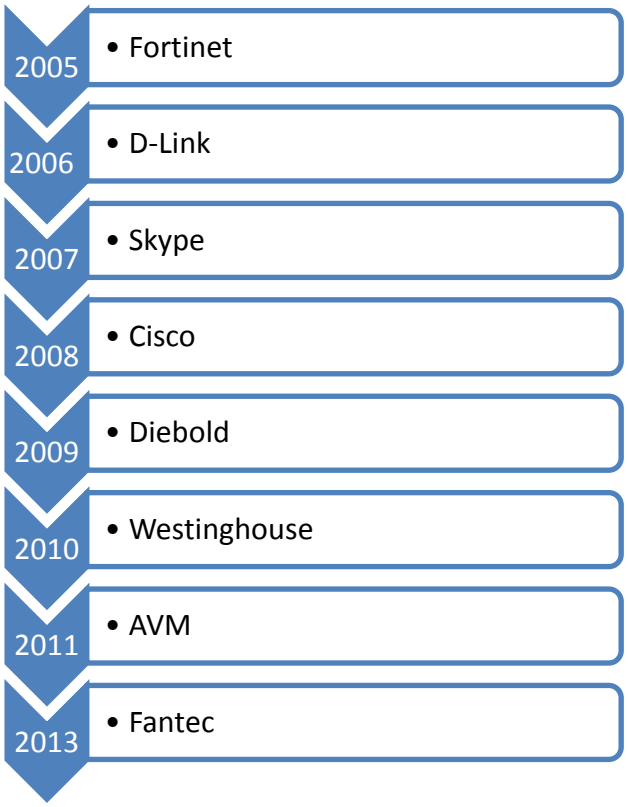

Abbildung 8 - Gerichtsverhandlungen bzgl. GPL-Lizenzen

Bei den Initiatoren der gerichtlichen sowie außergerichtlichen Durchsetzung der Einhaltung von Open Source Softwarelizenzbedingungen handelt es sich um Personen, die Open Source Softwarevereinigungen angehören, selbst an Open Source Softwareprojekten mitentwickelt haben oder auf andere Weise der Community angehören. Des Weiteren können auch Unternehmen Initiatoren außergerichtlicher Einigungen oder gerichtlicher Entscheidungen sein, wenn diese im Zusammenhang mit einem Open Source Softwareprodukt Wettbewerbsnachteile sehen.

Die Mehrheit an Streitfällen hinsichtlich der rechtmäßigen bzw. unrechtmäßigen Verwendung von Open Source Softwareprodukten wird allerdings außergerichtlich geregelt. So wurden durch das Projekt gpl.violations.org alleine im Jahr 2004 und 2005 mehr als 30 Streitfälle auf diese Weise beigelegt. (Ihlenfeld, www.golem.de, 2005)

Der Gründer des Projekts gpl-violations.org Harald Welte gibt 2006 an, er habe in den vergangenen 30 Monaten bei der Entdeckung und Verhandlung von mehr als 100 GPL-Verletzungen mitgewirkt und zahlreiche außergerichtliche Einigungen erzielt. (Welte H. , 2006) Zur Gründung des Projektes selbst führten zahlreiche bekannte GPL-Verletzungen, die Harald Welte, ein Miturheber eines Linux-Teilsystems, nicht weiter hinnehmen wollte. 2004 begann dieser daher neben außergerichtlichen Methoden auch gerichtliche Maßnahmen einzuleiten. Die FSF, von der Welte sich ungenügend unterstützt sah, folgte Weltes Beispiel erst 2008. (Jäger, 2008)

Folgende Übersicht gibt detaillierte Informationen zu genannten Präzedenzfällen wieder:

Jahr	**Gericht**	**Beklagter**	**Kläger**	**Urteil**
2005	LG München	Fortinet	Harald Welte, gpl-violations.org	Einstweilige Verfügung
2006	LG Frankfurt	D-Link	Harald Welte, gpl-violations.org	Auskunft und Kostenerstattung
2007	LG München	Skype	Harald Welte, gpl-violations.org	Unterlassung der Verbreitung
2008	US Gericht	Cisco	FSF	Einstellung des Verfahrens aufgrund außergerichtlicher Einigung
2010	US Gericht	Westinghouse	SFLC	Unterlassung und Schadensersatz
2011	LG Berlin	Cybits AG	AVM	Teilweiser Unterlassungsanspruch

2013	LG Hamburg	FANTEC	Harald Welte, gpl-violations.org	Auskunft und Kostenerstattung
2016	LG Hamburg	VMware	Christoph Hellwig	Klage abgewiesen

Tabelle 9 - Detaillierte Darstellung von Gerichtsverhandlungen bzgl. (L)GPL-Lizenzverletzungen

6.2.1 Fortinet (2005) – LG München

Gegen die britische Niederlassung des Softwareherstellers Fortinet, zu dessen Portfolio 2006 unter anderem Firewall- und Antivirenprogramme zählten, konnte am Münchner Landesgerichtshof eine einstweilige Verfügung erzielt werden. Die Klage wurde vom Projekt gpl-violations.com eingereicht, da das Unternehmen im Betriebssystem der vertriebenen Softwareprodukte weitere Softwareprodukte nutze, die unter der GPL stünden. Des Weiteren habe Fortinet sich geweigert die Nutzung dieser Softwareprodukte in ihrem Betriebssystem zu unterlassen und nicht auf außergerichtliche Einigungsversuche reagiert. Daher sei die Einreichung einer Klage unvermeidbar gewesen. Harald Welte, der Gründer des Projekts gpl-violations.com kritisiert in diesem Fall besonders, dass nicht nur die Lizenzbedingungen verletzt wurden, sondern mittels kryptografischer Maßnahmen bewusst versucht wurde, den Einsatz von unter der GPL befindlichen Softwareprodukten zu verschleiern. Welte habe außerdem kein Problem damit, dass Open Source Software kommerziell genutzt werde, man müsse sich aber an die Lizenzbedingungen halten. Konkret meinte Welte in diesem Zusammenhang die Offenlegung des Source Codes. Die erzielte einstweilige Verfügung formulierte sich konkret darin, dass Fortinet der Vertrieb der strittigen Software untersagt wurde, solange diese nicht den Lizenzbedingungen der GPL entspricht. (Ihlenfeld, www.golem.de, 2005) (Welte H. , 2005) (Wilkens, 2005)

6.2.2 D-Link (2006) – LG Frankfurt

Gegen die deutsche Tochtergesellschaft der Firma D-Link konnte erstmalig nach Bestätigung der Wirksamkeit der GPL-Lizenz durch den Entscheid des Landesgerichtshofs München zum Fall Fortinet eine Hauptsachenentscheidung erzielt werden. Eingereicht wurde die Klage ebenso wie im Fall Fortinet vom Gründer des Projekts gpl-violations.org Harald Welte, welcher D-Link den Vertrieb von Datenspeichern unterstellte, auf welchen sich Softwareprodukte befänden, die unter der GPL stünden. Welte betonte erneut, dass er sich über den vermehrten Einsatz von Open Source Softwareprodukten in kommerziellen Softwareprodukten freue, dennoch habe man sich an die Lizenzbedingungen zu halten. D-Link argumentierte einerseits mit kartellrechtlichen Nachteilen, die die GPL mit sich brächte und daher ungültig mache, und andererseits mit der Unzulässigkeit der Beweise, da diese nur mittels Reengineering erzielt werden könnten, was nicht zulässig sei. Mit dem Urteil des Landesgerichtshofs Frankfurt zur Kostenerstattung der Beweismittelerbringung und Offenlegung der Verkäufe, wurde gerichtlich verdeutlicht, dass die Lizenzbedingungen der GPL unabhängig von kartellrechtlichen Belangen einzuhalten sind, und

Reengineering zur Beweisführung zulässig ist, da ein Vergehen in solchen Fällen nicht anders bewiesen werden kann. (Ihlenfeld, 2006) (Koglin, 2006) (Welte, 2006)

6.2.3 Skype (2007) – LG München

Gegen das Luxemburger Unternehmen Skype Technologies SA, konnte am Münchner Landesgerichtshof im Anschluss an eine einstweilige Verfügung, ein Urteil auf Unterlassung bei der Mitwirkung des Vertriebs eines Telefons erzielt werden. Kläger Harald Welte warf dem Unternehmen, welches VoIP-Softwareprodukte anbot, vor, ein VoIP-Telefon der spanischen Firma SMC Networks zu vertreiben, welches die Lizenzbedingungen der GPL nicht erfülle. Dem besagten Telefon wurde lediglich ein Beiblatt beigelegt, in welchem darauf hingewiesen wurde, dass die Software des Telefons den Bestimmungen der GPL unterliegt. Außerdem wurde im Beiblatt lediglich auf einen Link verwiesen, unter welchem der Source Code der unter der GPL befindlichen Software abgerufen werden konnte. Skype Technologies SA argumentierte mit kartellrechtlichen Benachteiligungen, welchen zufolge die GPL nicht gültig sei. Des Weiteren wurde das Beilegen eines Informationsblattes mit Verweisen auf online-zugängliche Information als ausreichend empfunden. Der Landesgerichtshof München widersprach dieser Ansicht und entschied, dass die Art und Weise des VoIP-Telefons nicht zulässig sei. Konkret sei das Beilegen eines solchen Informationsblattes mit Verweisen auf Websites bei Hardware nicht zulässig. Dies wäre ausschließlich bei online vertriebener Software rechtskonform. Demzufolge ist sowohl der Source Code als auch der Lizenztext der GPL kostenfrei bei nicht online bezogener Software vollständig beizufügen. Neben dem Aspekt, dass erstmals gegen ein ausländisches Unternehmen geklagt und ein rechtskräftiges Urteil erzielt werden konnte, wurde die Gültigkeit der GPL und der zugehörigen Lizenzbedingungen erneut bestätigt und deren Auswirkungen konkretisiert. Mit dem Hersteller des VoIP-Telefons konnte eine außergerichtliche Einigung erzielt werden. (Diedrich, heise.de, 2007) (Ihlenfeld, golem.de, 2007) (Küng, 2008)

6.2.4 Cisco (2008) – US Gericht

Gegen das auf Netzwerktechnik spezialisierte Unternehmen Cisco Systems Inc., wurde an einem New Yorker Gericht Klage eingereicht. Die FSF warf dem Unternehmen Verletzungen der Lizenzbedingungen einiger Programme des GNU Projektes vor, welche unter der GPL stehen. Aufgrund des nicht vollständig zur Verfügung gestellten Source Codes, forderte die FSF sowohl Unterlassung als auch Schadensersatz. Zuvor getätigte Bemühungen seitens der FSF, den Fall außergerichtlich zu lösen, blieben fruchtlos. Dieser Fall stellte die erste gerichtliche Klage durch die FSF dar, da diese zuvor nur außergerichtliche Schritte in Erwägung zog. (Jäger, 2008) 2009 wurde das Verfahren jedoch eingestellt, da es den Prozessparteien nun doch möglich war, eine außergerichtliche Einigung zu treffen. Cisco erklärte sich zur Begleichung eines Schadensersatzes in nicht offiziell bekannter Höhe bereit, und verpflichtete sich zur Einhaltung der GPL. Dies wiederum sollte durch eine eigens dafür ernannte Person bei Cisco umgesetzt werden, deren Maßnahmen und Erfolge an die FSF zu kommunizieren waren. Auch der fehlende vollständige Source Code zu den strittigen Produkten sollte von Cisco auf deren Website nun zur Verfügung gestellt werden. (Kueng, 2009)

6.2.5 Westinghouse (2010) – US Gericht

Gegen den Elektronikhändler Westinghouse Digital Technologies LLC konnte 2010 an einem New Yorker Gericht ein Urteil auf Unterlassung und Schadenersatz erzielt werden. Das Software Freedom Law Center (SFLC) unterstellte dem Unternehmen Urheberrechtsverletzung, da dieses Fernsehgeräte auf eine der GPL widersprechenden Art und Weise vertrieb. Konkret handelte es sich dabei um die Verwendung der unter der GPL befindlichen Toolsammlung Busybox, welche in einigen vertriebenen Fernsehgeräten integriert war. Den aus der GPL abzuleitenden Bedingungen, den Source Code sowie die Lizenzbedingungen zur Verfügung zu stellen, wurde dabei nicht nachgekommen. Der Elektronikhändler erschien jedoch nicht zur Gerichtsverhandlung, sondern meldete stattdessen Insolvenz an. Dennoch, oder gerade deshalb, wurde den Klägern Recht gegeben, und das Unternehmen auf Unterlassung des Vertriebs samt Schadenersatzleistung und Übernahme der Gerichtskosten verurteilt. Mit diesem Urteilsspruch konnte erstmalig in den USA die Gültigkeit der GPL gerichtlich verdeutlicht werden, da bislang nur außergerichtliche Einigungen erzielt werden konnten. (Lindner, 2010) (Müller, 2010) (Roger, ifross.org, 2010)

6.2.6 AVM (2011) – LG Berlin

Anders als in den bereits beschriebenen Beispielen klagte in diesem Fall der Router Hersteller AVM den Anbieter von Filtersoftware Cybits AG, wodurch AVM diesem und allen anderen Unternehmen die Veränderung der Firmware der Fritzbox, ein eigener Router welcher von AVM hergestellt und vertrieben wurde, untersagen wollte, obwohl die Firmware des Routers auf Linux basierte, welche den Bedingungen der GPL unterliegt. Konkret warf AVM Cybits urheberrechtliche- sowie markenrechtliche Verletzungen vor, wenn deren Kunden deren Filtersoftware auf Fritzbox-Routern von AVM installierten. Dieser Fall war von besonderem Interesse, da bei erfolgreicher Klage durch AVM gegen die Grundsätze freier Software entschieden worden wäre und demzufolge Unternehmen anderen untersagen hätten dürften, freie Software zu verändern. (Ihlenfeld, golem.de, 2011) Das Gericht widersprach allerdings den Forderungen von AVM und bestätigte entsprechend der Grundsätze freier Software, dass AVM keine grundlegenden ureberrechtlichen oder markenrechtlichen Ansprüche gegen Cybits und anderen Unternehmen stellen kann. Einen Teilerfolg konnte AVM dennoch erzielen, da die Software von Cybits bei Installation fehlerhafte Anzeigen direkt im Interface der Fritzbox auslöste. Da dies von Kunden als Produktfehler der Fritzbox, und nicht als Fehler der Filtersoftware wahrzunehmen war, stellte dies für AVM einen wettbewerbsrechtlichen Nachteil dar, wodurch AVM gegen Cybits einen Unterlassungsanspruch erzielen konnte. Mit diesem Urteil bestätigte das Landesgericht Berlin erneut die Gültigkeit der GPL, und definierte genauer, dass die Veränderung von unter der GPL befindlicher Firmware durch andere zulässig ist. Dies darf allerdings nicht zu wettbewerbsrechtlichen Nachteile anderer Hersteller führen. (Roger, ifross.org, 2011)

6.2.7 FANTEC (2013) – LG Hamburg

Gegen das Hamburger Unternehmen Fantec, ein Hersteller von Mediaplayern, konnte am Landesgerichtshof Hamburg 2013 ein Urteil erzielt werden. Das Unternehmen war Hersteller und Vertreiber eines Mediaplayers, dessen Firmware unter anderem der GPL unterliegt. Harald Welte, dessen Bemühungen eine außergerichtliche Einigung zu erzielen zuvor erfolglos blieben, warf Fantec vor, im Rahmen des Online-Vertriebs des Mediaplayers die Lizenzbedingungen der GPL nicht eingehalten zu haben. Das Unternehmen habe einerseits nicht auf die GPL und deren Lizenzbedingungen hingewiesen, und später einen Source Code veröffentlicht, welcher einen älteren Code Stand aufwies, als der Source Code aus denen das tatsächliche Binärprogramm abgeleitet wurde. Fantec argumentierte von dem älteren Source Code Stand nichts gewusst zu haben, da dessen Aktualität von einem eigenen Zulieferer, auf welchen die Entwicklung der Firmware ausgelagert wurde, versichert wurde. Das Landesgericht Hamburg beschrieb in seinem Urteil diese Vorgangsweise allerdings als fahrlässig, da Fantec selbst für die Richtigkeit, Vollständigkeit und Aktualität von veröffentlichten Quellen Sorge zu tragen hat. Das Unternehmen wurde daher zu Zahlung von Schadenersatz, Kostenübernahme, sowie zu detaillierter Auskunft über die mit dem Tatbestand verbundenen Vertriebswegen, verurteilt. Entdeckt wurde die Verletzung der GPL 2012 im Rahmen eines Workshops unter dem Namen „Hacking for Compliance“, bei dem Freiwillige die zugrundeliegende Software diverser Geräte auf deren Lizenzen und die damit verbundene Einhaltung der Bedingungen überprüften. Harald Welte, und seinem Projekt gpl-violations.org, wurden die Ergebnisse des Workshops im Anschluss übergeben, worauf dieser entsprechende rechtliche Schritte einleitete. (Dölle, 2013) (FSFE, 2013)

6.2.8 VMware (2016) – LG Hamburg

Die Klage gegen den Virtualisierungsexperten VMware aufgrund von Verletzung der GPL wurde am Hamburger Landesgericht in erster Instanz abgewiesen. Strittig war das von VMware hergestellte und vertriebene Softwareprodukt ESXI, ein Virtualisierungsprodukt. Dieses besteht aus dem proprietären VM-Kernel, welches über eine eigene API mit dem VMK-Linux kommuniziert. VMware bestätigte, dass VMK-Linux eine Ableitung des Linux-Kernels ist, wodurch dieser Teil auch unter der GPL lizenziert und entsprechend frei zur Verfügung gestellt wurde. Der Kläger Christoph Hellwig, welcher selbst mehrere Jahre an Teilen des Linux-Kernels mitgearbeitet hatte, widersprach allerdings der Sichtweise VMwares. Ihm zufolge seien der VM-Kernel, VMK-Linux, sowie die dazugehörige API als eine untrennbare Einheit zu betrachten, da die Bestandteile nicht eigenständig nutzbar seien. Demzufolge dürfe der VM-Kernel nicht proprietär vertrieben werden, sondern müsse ebenfalls unter der GPL lizenziert werden und frei zur Verfügung stehen. Thematisiert wurde unter anderem auch, ob die Beiträge an der Linuxentwicklung, die von Hellwig stammten, für diesen Fall relevant waren. Hellwig habe an seinen entwickelten Teilen ein Bearbeitungsurheberrecht, wodurch er genau nachweisen müsse, welche von ihm entwickelten Codestellen in welcher Art und Weise von VMware in VMK-Linux übernommen und genutzt werden. Zur Klärung, ob Hellwigs Bearbeitungen VMware zufolge vernachlässigbar oder relevant seien, würde ein Sachverständiger benötigt werden. In erster

Instanz wurde die Klage Hellwigs abgewiesen, da dieser dem Gericht zufolge die Relevanz seiner Bearbeitung in VMK-Linux nicht konkret genug nachweisen konnte. Auf die Kernthematik, ob es sich bei den beschriebenen Bestandteilen von ESXI um getrennt zu betrachtende Teile oder eine Einheit handelt, wurde im Urteil nicht eingegangen. Hellwig kündigte Berufung an. (Diedrich O. , 2016) (Labesius, 2016)

6.3 Fazit

Das Modell der Open Source Softwareentwicklung birgt auch für Unternehmen Vorteile und liefert Anreize diese zu unterstützen und voranzutreiben. Dadurch profitieren Open Source Softwareprojekte ebenso durch die Beteiligung von Unternehmen. Dabei gibt es unterschiedliche Szenarien, wie sich Unternehmen das Modell der Open Source Softwareentwicklung zu Nutze machen können. Die Beteiligung kann durch Weiterentwicklung bestehender Open Source Softwareprodukte, Initiierung neuer Open Source Softwareprodukte oder durch Freigabe ehemals proprietärer Softwareprodukte unter einer Open Source Softwarelizenz geschehen.

Das Unternehmen selbst kann dabei unterschiedliche Positionierungsstrategien verfolgen, indem es Open Source Softwareprodukte als Hauptgeschäftszweig erzeugt, Komplemente zu Open Source Software herstellt, Open Source Software für interne Prozesse verwendet, oder Open Source Softwareprodukte in eigene proprietäre Softwareprodukte einbaut. Je nach Positionierung können zu den bereits bekannten Vor- und Nachteilen weitere mögliche Vor- und Nachteile für Unternehmen auftreten. Mögliche Vorteile können Steigerung des Marktanteils, Möglichkeit der Querfinanzierung durch Komplemente, positive Reputation, Behebung von Sicherheitslücken sowie Ressourcenersparnisse darstellen. Denen gegenüber stehen mögliche Nachteile wie Lizenzbindung, Investitionsunsicherheit, Wettbewerbsverluste, sowie Einschränkungen hinsichtlich der möglichen Integrierbarkeit in proprietäre Softwareprodukte und damit verbundene potentielle Gefährdungsmöglichkeiten hinsichtlich des Geschäftsmodells des Unternehmens.

In Bezug auf die Beantwortung der Forschungsfrage ist jene Positionierungsmöglichkeit relevant, bei der Unternehmen Open Source Softwareprodukte in ihre eigenen, proprietären Softwareprodukte integrieren. Aufgrund der Erläuterungen der vorangegangenen Kapitel ergibt sich, dass Open Source Softwareprodukten mit strengen Copyleft nicht in proprietäre Softwareprodukte eingebunden werden dürfen, wenn diese proprietär bleiben sollen. Auch die Einbindung von Open Source Softwareprodukten mit beschränktem Copyleft bringt Auflagen mit sich, die eingehalten werden müssen, um die Lizenzbedingungen nicht zu verletzen. Dementsprechend beschäftigen sich Gerichte und Interessensvertreter der Open Source Community mit Fällen, die die Einbindung von Open Source Softwareprodukten unter den genannten Open Source Softwarelizenzen in proprietäre Softwareprodukte betreffen. Die Mehrheit an Streitfällen hinsichtlich der rechtmäßigen bzw. unrechtmäßigen Verwendung von Open Source Softwareprodukten wird allerdings außergerichtlich geregelt.

Aus der Analyse der behandelten Präzedenzfälle ergibt sich, dass die klagende Partei zumeist aus dem Umfeld der Open Source Softwareentwicklung stammt. Genauer können dies Softwareentwicklerinnen und Softwareentwickler sein, die selbst an der Entwicklung eines Open

Source Softwareprojektes mitgewirkt haben und die Einhaltung der Lizenzbedingungen des entsprechenden Softwareprodukts durchsetzen wollen, oder die Vereinigungen und Organisationen, die hinter dem Modell der Open Source Softwareentwicklung stehen, bilden die klagende Partei. Um die Verletzungen überhaupt aufdecken zu können werden Methoden des Reverse Engineerings angewandt, um den Einsatz des betreffenden Open Source Softwareprodukts im proprietären Softwareprodukt und die damit verbundenen Lizenzverletzungen beweisen zu können. Die Durchführung des Reverse Engineerings wird wiederum von denselben Parteien veranlasst, die sich für das Modell der Open Source Softwareentwicklung einsetzen. Erfolgsfaktor dieser Methodik sind wiederum die sozialen Bedürfnisse der Softwareentwicklerinnen und Softwareentwickler, die in Kapitel 6.1 beschrieben wurden. Auch werden von den Interessensvertretern die hinter dem Open Source Modell stehen Workshops und weitere Treffen mit Freiwilligen aus der Community ins Leben gerufen, um gemeinschaftlich proprietäre Softwareprodukte auf mögliche Lizenzverletzungen zu prüfen. Eine weitere Klägergruppe stellen Unternehmen dar, die Wettbewerbsnachteile erfahren, die durch Komplemente anderer Hersteller verursacht werden.

Zu Verhandlungen an europäischen und US-amerikanischen Gerichten kam es, da sich einige Unternehmen weigerten, mit den Interessensvertreten des Open Source Modells außergerichtliche Einigungen zu erzielen. Durch die dabei erzielten Urteile konnte die Wirksamkeit der Open Source Lizenzbedingungen bestätigt und weiter ausgeführt werden. Unternehmen haben daher bei Verurteilung mit Unterlassungsansprüchen, Kostenerstattung, Offenlegung der Vertriebsdetails sowie mit Schadenersatzleistungen zu rechnen. Die mehrfach bezogene Position der beklagten Unternehmen, die Bedingungen der Open Source Lizenzen führen zu Wettbewerbsnachteilen und seien daher nicht gültig, wurde durch die erzielten Urteile von den Gerichten ebenso als nicht relevant tituliert.

7 ERARBEITUNG EINES OPEN SOURCE GENEHMIGUNGSPROZESSES

Die Ergebnisse der vorangegangenen Kapitel haben gezeigt, dass der Einsatz von Open Source Softwareprodukten mit strenger Copyleft Klausel nicht in proprietäre Softwareprodukte eingebunden werden dürfen, wenn diese proprietär bleiben sollen. Auch die Einbindung von Open Source Softwarelizenzen mit beschränkter Copyleft Klausel bringt Auflagen mit sich, die eingehalten werden müssen. Des Weiteren muss das Unternehmen sicherstellen können, dass die Bearbeitungslinie aller Weiterentwicklungen des eingesetzten Open Source Softwareprodukts genau nachvollzogen werden kann, da das Unternehmen im Streitfall sein korrektes Vorgehen beweisen können muss. Problematisch können daher auch Lizenzverletzungen in vorangegangenen Bearbeitungen des eingesetzten Open Source Softwareprodukts sein, wenn in diese beispielsweise proprietärer Source Code eingebaut wurde, oder Source Code von Softwareprodukten mit strengem oder beschränktem Copyleft. Um den rechtmäßigen Umgang mit Open Source Softwareprodukten sicherzustellen, arbeiten die Vereinigungen, welche sich für das Modell der Open Source Software einsetzen, hart daran, Lizenzverletzungen aufzudecken. Dabei kommt es in den meisten Fällen zu außergerichtlichen Einigungen, in den letzten zehn Jahren gab es dennoch einige gerichtliche Urteile, die die Grundsätze des Open Source Softwaremodells und ihre Interessensvertreter bestätigten. Verletzen Unternehmen die Lizenzbedingungen der eingebundenen Open Source Softwareprodukte, ist laut den bisher getroffenen gerichtlichen Entscheidungen mit Unterlassungsansprüchen, Kostenübernahme, Schadenersatzleistungen sowie Offenlegung von Vertriebsdaten zu rechnen. Da es sich bei den meisten proprietären Softwareprodukten um kommerzielle Softwareprodukte handelt, durch dessen Verkäufe die Umsätze und Gewinne des Unternehmens erzielt werden sollen, kann die unrechtmäßige Einbindung eines Open Source Softwareprodukts in das kommerzielle, proprietäre Softwareprodukt das Geschäftsmodell des Unternehmens gefährden. Dabei muss die Lizenzverletzung durch das Unternehmen nicht einmal vorsätzlich durchgeführt worden sein, es reicht, wenn es Verletzungen in der Bearbeitungskette gibt. Dennoch bieten Open Source Softwareprodukte viele Vorteile, wie die Einsparung von Ressourcen, wodurch sich manche Geschäftsmodelle erst realisieren lassen. Daher ist es für Unternehmen, die bereits Open Source Softwareprodukte in Verbindung mit ihrem proprietären Softwareprodukt einsetzten bzw. in Zukunft einsetzen wollen, wichtig, einen Prozess zu schaffen, mit dem es möglich ist, den Einsatz von Open Source Softwareprodukten kontrolliert zu steuern. Ebenso müssen Mitarbeiterinnen und Mitarbeiter über den richtigen Umgang mit Open Source Softwareprodukten in Bezug auf das Geschäftsmodell des Unternehmens, für das sie tätig sind, Bescheid wissen und die Richtlinien des Unternehmens mittragen.

7.1 Rahmen des Anwendungsfalls

Der Open Source Genehmigungsprozess wurde im Rahmen der Produktentwicklungsabteilung eines Softwareentwicklungsunternehmens umgesetzt in welcher ein kommerzielles, proprietäres Softwareprodukt hergestellt wird. Dafür wurden Expertinnen und Experten des Unternehmens vor und während der Definition des Prozesses befragt, um die Interessen des Unternehmens mit dem erarbeiteten Prozess wahren zu können.

7.1.1 Das Softwareprodukt

Bei dem kommerziellen, proprietären Softwareprodukt handelt es sich um ein Front- und BackOffice Business Support System, mit dessen Komponenten die Bereiche des Geschäftsprozessmanagements, des Kundenmanagements, des Produktportfoliomanagements, der Auftragsverwaltung sowie der Verrechnungs- und Finanzverwaltung abgedeckt werden können. Diese digitale Ökosystem Management Plattform ist technologie- und industrieunabhängig, und kann demnach von Unternehmen in den verschiedensten Sektoren eingesetzt werden, wie beispielsweise in den Bereichen Telekommunikation, Energie, Medien und Finanzservices. Das Softwareprodukt ist multimandantenfähig, wobei jeder einzelne Mandant die volle Funktionalität des Business Support Systems nutzen kann. Durch die Architektur des Softwareprodukts ist es möglich, verschiedene Services einzeln oder gebündelt, über verschiedene Verkaufskanäle zu unterschiedlichen Preiskonzepten an sowohl Endkunden, als auch anderen Mandanten auf der digitalen Plattform zu verkaufen.

7.1.2 Das Unternehmen

Die Kernkompetenzen des Unternehmens liegen im Bereich der hochtechnologischen IT Lösungen. Das Unternehmen beschäftigt ca. 400 Mitarbeiterinnen und Mitarbeiter, und ist eine hundertprozentige Tochter des zugehörigen Mutterkonzerns.

7.2 Expertinnen- und Expertenbefragung

Für die Erarbeitung des Open Source Genehmigungsprozesses sowie des damit verbundenen Leitfadens für den richtigen Einsatz von Open Source Softwareprodukten in proprietären Softwareprodukten, werden Expertinnen- und Experten befragt. Durch diese Befragung soll sich einerseits der Ist-Zustand hinsichtlich des Umgangs mit Open Source Softwareprodukten innerhalb des Unternehmens erheben lassen, sowie der Soll-Zustand definiert werden können. Des Weiteren sollen durch die Befragung weitere Faktoren erhoben werden können, die bislang nicht dokumentiert wurden.

7.2.1 Interviewgesprächsleitfaden

Für die Interviews mit den Expertinnen und Experten wurde im Vorfeld anhand der theoretischen Hintergründe in Bezug auf die Forschungsfrage folgender Interviewgesprächsleitfaden festgelegt, um die Befragung strukturiert durchführen zu können. Des Weiteren dient der Gesprächsleitfaden dazu, die Ergebnisse der Interviews zu bündeln und vergleichen zu können.

Der Interviewgesprächsleitfaden wurde wie folgt aufgebaut:

1. Einführung in das Themengebiet – *basierend auf Kapitel 1*

   a. Aufgabenstellung und Forschungsfrage
   b. Ziel der Befragung
   c. Einschränkung des Rahmens

2. Vorstellung der Expertin bzw. des Experten

   a. Name und Ausbildung
   b. Beschreibung der Rolle im Unternehmen und der damit verbundenen Aufgaben, Kompetenzen und Verantwortlichkeiten
   c. Beruflicher Bezug zum Themengebiet
   d. Eventuell persönlicher/privater Bezug zum Themengebiet

3. Lizenzierung des proprietären Softwareprodukts

   a. Welches Geschäftsmodell verfolgt das Unternehmen mit ihrem proprietären Softwareprodukt
   b. Wie wird das Softwareprodukt lizenziert? – *in Anlehnung an Kapitel 5.3*

4. Open Source Software in Unternehmen

   a. Welche Positionierung bezieht das Unternehmen in Bezug auf Open Source Software? – *in Anlehnung an Kapitel 7.1*
   b. Welche Open Source Softwareprodukte werden eingebunden?
   c. Unter welchen Lizenzen stehen die eingebundenen Lizenzen? – *in Anlehnung an Kapitel 6.3*
   d. Wie wird die Aufgabe des Lizenzmanagements wahrgenommen? – *in Anlehnung an Kapitel 3.1*
   e. Welche Vorteile und Chancen sieht die Expertin / der Experte im Einsatz von Open Source Softwareprodukten in kommerziellen, proprietären Softwareprodukten? – *in Anlehnung an Kapitel 6.5.1 und 7.1.5*
   f. Welche Nachteile und Risiken sieht die Expertin / der Experte im Einsatz von Open Source Softwareprodukten in kommerziellen, proprietären Softwareprodukten? – *in Anlehnung an Kapitel 6.5.1, 7.1.5 und 7.2*

   g. Welche Maßnahmen werden ergriffen, um Risiken vorzubeugen? – *in Anlehnung an Kapitel 7.2*
   h. Welche Eintrittswege für Open Source Softwareprodukte gibt es innerhalb der des Unternehmens in Bezug auf das proprietäre Softwareprodukt?
   i. Wie erfolgt die Prüfung und Freigabe?
   j. Wie erfolgt die technische Analyse des proprietären Softwareprodukts in Hinblick auf die Vermeidung von Risikopotentialen im Zusammenhang mit Open Source Softwareprodukten?
   k. Wie erfolgt die Dokumentation der eingesetzten Open Source Softwareprodukte?

5. Erarbeitung des Open Source Genehmigungsprozess
   a. Wie fasst die Expertin / der Experte die IST-Situation zusammen?
   b. Welcher Soll-Zustand soll aus Sicht der Expertin / des Experten erreicht werden?
   c. Welche Schritte sind zur Erreichung des Soll-Zustandes im zu erarbeitenden Open Source Genehmigungsprozess zu berücksichtigen?
   d. Welche anderen Prozesse des Unternehmens können / sollen auf den Open Source Genehmigungsprozess Einfluss haben?
   e. Welche anderen Prozesse des Unternehmens kann / soll der Open Source Genehmigungsprozess beeinflussen?
   f. Welche Rollen sind in Bezug auf ihre Aufgaben, Kompetenzen und Verantwortlichkeiten vom Open Source Genehmigungsprozess betroffen? In welchem Ausmaß?
   g. Welche Inhalte soll der Leitfaden laut Expertin bzw. Experten beinhalten, um für Mitarbeiterinnen und Mitarbeiter eine nützliche Hilfestellung darzustellen?
6. Folgeschritte
   a. Welche Maßnahmen sind aus Sicht der Expertin bzw. des Experten beim Ausrollen des Open Source Genehmigungsprozesses durchzuführen?
   b. Welche Folgeschritte sind aus Sicht der Expertin bzw. des Experten im Anschluss an das Ausrollen des Open Source Genehmigungsprozesses sinnvoll bzw. notwendig?

7.2.2 Reviewgesprächsleitfaden

Zuzüglich der Interviewtermine wurden mit den Expertinnen und Experten zwei Reviewtermine vereinbart, um sicherstellen zu können, dass die Interessen der jeweiligen Expertin bzw. des jeweiligen Experten im Open Source Genehmigungsprozess sowie in der Erstellung des Leitfadens hinsichtlich des Umgangs mit Open Source Softwareprodukten, vertreten sind.

Der Reviewgesprächsleitfaden für das erste Reviewgespräch wurde wie folgt aufgebaut:

1. Vorstellung der gesammelten Interessen aller befragten Expertinnen und Experten
2. Vorstellung des Zwischenergebnisses
3. Verweis auf Realisierung der geäußerten Interessen, die von der jeweiligen Expertin bzw. dem jeweiligen Experten geäußert wurden
4. Inhaltliche Diskussion der Umsetzung
5. Festhalten von Verbesserungspotentialen

Der Reviewgesprächsleitfaden für das zweite Reviewgesprächs wurde wie folgt aufgebaut:

1. Vorstellung der gesammelten Verbesserungspotentiale aller befragten Expertinnen und Experten
2. Vorstellung des überarbeiteten Zwischenergebnisses
3. Verweis auf die Realisierung der durch die Expertin bzw. den Experten geäußerten Verbesserungswünsche
4. Inhaltliche Diskussion der Umsetzung
5. Festhalten von Verbesserungspotentialen (optional)

Die Finale Freigabe sollte durch die Abteilungsleitung der Produktentwicklungsabteilung erfolgen, da der Prozess innerhalb dieser Abteilung angewendet werden sollte.

7.2.3 Expertinnen und Experten

Folgende Expertinnen und Experten wurden für die Befragung hinsichtlich Open Source Software in kommerziellen, proprietäre Softwareprodukten und der daraus resultierenden Erstellung eines Open Source Genehmigungsprozesses ausgewählt:

Expertin / Experte	**Rolle / Aufgabe im Unternehmen**
Person 1	Qualitätssicherung und Prozessmanagement in der Abteilung der Produktintegration
Person 2	Produktmanagement innerhalb der Produktentwicklungsabteilung
Person 3	Qualitätssicherung in der Produktentwicklungsabteilung
Person 4	Leitung der Produktentwicklungsabteilung
Person 5	Technologie-Innovation in der Produktentwicklungsabteilung
Person 6	Erstellung und Koordination der technischen Dokumentation
Person 7	Softwarearchitektur in der Produktentwicklungsabteilung

Tabelle 10 - Auflistung der befragten Expertinnen und Experten

In Zusammenhang mit dieser Masterarbeit wurden die befragten Expertinnen und Experten anonymisiert und geschlechterneutral mit Person 1-7 bezeichnet. Die Auswahl von Expertinnen

und Experten aus unterschiedlichen Unternehmensbereichen mit unterschiedlichen Aufgaben, Kompetenzen und Verantwortlichkeiten zielt darauf ab, die verschiedenen Sichtweisen und Interessen der Unternehmensbereiche in der Umsetzung des Open Source Genehmigungsprozesses abzudecken.

Die Interviewtermine sowie die Termine zum Review wurden zeitlich gebündelt und in kurzen Intervallen vereinbart, um die unterschiedlichen Sichtweisen ebenso gebündelt in die Erstellung des Open Source Genehmigungsprozesses einfließen lassen zu können. Genauer fand anschließend an die Erstbefragung der Expertin bzw. des Experten das erste Reviewgespräch nach einer Woche statt, worauf wiederum innerhalb einer Woche bei Bedarf das zweite Review durchgeführt wurde. Dadurch konnte den Expertinnen und Experten ebenso die Sichtweise der anderen Expertinnen und Experten vermittelt werden, wodurch innerhalb dieses Personenkreises eine gemeinsame Sichtweise erreicht werden konnte. Die Zustimmung zur Anwendung des Prozesses innerhalb der Produktentwicklungsabteilung erfolgte von allen Expertinnen und Experten, wobei die finale Freigabe durch Experte 4 erfolgte, da dieser die Abteilungsleitung der betreffenden Abteilung innehat.

7.3 Erhebung des IST-Zustandes

Aufgrund der durchgeführten Expertinnen- und Expertenbefragung, ergab sich folgende IST-Situation:

7.3.1 Vorteile von Open Source Softwareprodukten

Die Aussage*, „das Rad nicht neu erfinden zu wollen“* äußerten sowohl Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung), 3 als auch 4, wobei letztere ergänzt, dass sie im Modell der Open Source Softwareentwicklung ebenso technische Vorteile sieht, ohne die die Herstellung eines Softwareprodukts, das des Unternehmens in vergleichbarer Form nicht möglich wäre. Ebenso erkennt Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) das Einsparen von Ressourcen als Vorteil an.

Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) führt näher aus, dass mittels Open Source Software generische technische Lösungen eingesetzt werden können, die in vielen Unternehmen benötigt werden. Dadurch kann man sich auf das eigene Business konzentrieren und muss sich nicht auf die Entwicklung von Basiskomponenten konzentrieren. Demzufolge spart man sich ebenso die Wartungsaufwände für Basiskomponenten. Auch Security Probleme werden in Open Source Softwareprodukten aufgedeckt und gelöst, ohne dass man dies selbst durchführen muss.

Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung) äußert, dass neben dem Einsatz von vorhandenen Technologien auch die Absicherung der Qualität des Source Codes als Vorteil anzusehen ist, wenn die entsprechende Community hinter der gewählten Technologie

vorhanden ist. Deren Stärke wird bei der Evaluierung eines Open Source Softwareprodukts mitberücksichtigt und ist entscheidender Faktor. Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) ergänzt, dass bei starker Community davon auszugehen ist, dass das Open Source Softwareprodukt potentiell mehr Anwendungsfälle abdeckt und ebenso breit getestet ist. Dadurch ist Open Source Software um ein Vielfaches robuster, als man es bei Eigenproduktion gewährleisten könnte. Auch die Komplexität mancher Open Source Softwareprodukte wäre intern nicht handhabbar. Als Firma in der Größenordnung des Unternehmens führt Person 4 (Leitung der Produktentwicklungsabteilung) näher aus, wäre es mittels Innovationsbudget nicht möglich, ein vergleichbares Grundlagenwissen wie das der Community aufzubauen.

Allgemein äußert Person 4 (Leitung der Produktentwicklungsabteilung), dass man das Modell der Open Source Softwareentwicklung mit Standards der Industrie, des Maschinenbaus oder der Elektronik vergleichen kann. Hier gibt es Standardkomponenten, die eingekauft werden und mit denen eine Maschine oder ein elektronisches Gerät gebaut wird. Dies, bezogen auf die Abhängigkeit von Lieferketten, wurde in der Softwareentwicklung erst mit dem Modell der Open Source Softwareentwicklung möglich, und demzufolge das gemeinschaftliche Entwickeln von komplexen Systemen. Die Existenz des Open Source Softwaremodells ist die logische Folge des Fehlens von genormten Standardkomponenten.

Als weiteren Vorteil sehen Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) und 7, dass Eigenentwicklungen als Open Source Softwareprodukt veröffentlicht werden können um für positive Reputation und Gewinnung neuer Mitarbeiter zu sorgen. Ebenso sind große Open Source Projekte in Kreisen der Softwareentwicklerinnen und Softwareentwickler bekannt, was wiederum attraktiv für die Gewinnung neuer Mitarbeiterinnen und Mitarbeiter sein kann.

7.3.2 Nachteile von Open Source Softwareprodukten

Als potentielles Risiko im Zusammenhang mit der Integration von Open Source Software in das proprietäre Softwareprodukt des Unternehmens werden von allen Expertinnen und Experten Lizenzverstöße gesehen, die zu Klagen führen können.

Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) erwähnt in diesem Zusammenhang, dass ebenso Kunden potentielle Kläger darstellen, da diese oft selbst über Tools verfügen, um erworbene Softwareprodukte zu testen, oder Nachweise darüber verlangen, dass Open Source Software entsprechend derer Lizenzbedingungen integriert wurde. Werden hierbei Lizenzverstöße aufgedeckt, ist dies auch rufschädigend für das Unternehmen. Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) führt aus, dass aus ihrer Sicht rechtliche Risiken ein schier unlösbares Problem darstellen, da die Thematik sehr komplex ist und teilweise Interpretationsspielraum zulässt. Dies sei die größte Hürde für Unternehmen, Open Source Softwareprodukte in ihr eigenes proprietäres Softwareprodukt einzubinden. Vor allem für kleine Unternehmen ist dies ein Problem, da diese entgegen großen Unternehmen und Konzernen nicht über eine eigene Rechtsabteilung verfügen.

Person 4 (Leitung der Produktentwicklungsabteilung) äußert, dass die Problematik beim Kopieren und Einfügen von Source Code der komplizierte rechtliche Nachweis darüber ist, wem das Open Source Softwareprodukt gehört, wer es erfunden hat, und in wie weit der- oder diejenige es erfunden hat. Da es sich um einen kreativen Prozess handelt ist das Urheberrecht immer ein Thema. Die Leute, die handeln, sind meist technisch fokussiert und setzen sich kaum mit den rechtlichen Rahmenbedingungen auseinandersetzen. Ebenso erwähnt sei, dass auch Juristen noch zu wenig Erfahrung in der Handhabung von Open Source Themen haben, da die rechtliche Problematik erst in den letzten 15 Jahren an Gerichten behandelt wurde. Unabhängig davon stellen der entstandene Wildwuchs an Open Source Lizenzmodellen und deren Bedingungen ein großes Risiko für Unternehmen dar, die Open Source Softwareprodukte in ihre proprietären Softwareprodukte einbinden wollen.

Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) erwähnt, dass Open Source Software nicht automatisch mit guter Software gleichzusetzen ist. Daher ist darauf zu achten, dass man ein weit verbreitetes Open Source Softwareprodukt verwendet, welches entsprechend getestet, stabil und ausgereift ist. Zu weiteren Risiken zählen das Verweisen von Projekten sowie Forking, wodurch die kritische Menge an Entwicklern für die jeweiligen verbleibenden Projekte zu gering werden könnte. Ebenso kann bei Konzentration auf wenige Kernpersonen, das Ausscheiden einer Person aus dem Projekt das Einstellen des Projektes bedeuten. Dies bestätigt Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung), da es keine Verträge gibt, wodurch man auch kein Anrecht darauf hat, dass ein Open Source Projekt weiterverfolgt wird.

Mögliche unterschiedliche Anforderungen zwischen denen, die das Open Source Projekt verfolgt, und denen, die das Projekt verfolgt, in welches das Open Source Produkt eingebunden werden soll, stellen laut Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) und 7 ebenso ein weiteres Risikopotential dar.

Ebenso betont Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung), dass Upgrades von bereits eingebundenen Open Source Softwareprodukten, Lizenzänderungen zur Folge haben können. Dies bestätigt Peron 5, die beschreibt, dass gerade durch Upgrades problematische Softwarebibliotheken in das Produkt gelangen können, wenn sich die dahinterliegenden Abhängigkeiten verändern.

7.3.3 Lizenzierung des proprietären Softwareprodukts

Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) und 4 erläutern, dass es sich bei dem Softwareprodukt um ein kommerzielles, proprietär vertriebenes Softwareprodukt handelt, dass einerseits durch Lizenzverkäufe, und andererseits durch spezifische Anpassung an den Kunden Gewinne für das Unternehmen einbringt, welche wiederum unter anderem der Finanzierung der Weiterentwicklung des Standardprodukts dienen sollen. Des Weiteren werden Serviceleistungen wie Wartungsverträge angeboten. Person 4 (Leitung der Produktentwicklungsabteilung) ergänzt, dass die Entwicklung des Standardproduktes und die Integration des angepassten Standardproduktes beim Kunden deshalb getrennt zu betrachten sind, da die Integration auch von Partnerunternehmen durchgeführt werden kann.

7.3.4 Open Source Software im Unternehmen

Alle Expertinnen und Experten geben an, dass innerhalb des Unternehmens Open Source Softwareprodukte sowohl für interne Prozesse, als auch innerhalb des proprietären Softwareprodukts verwendet werden.

Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) gibt an, dass sie für ihre Tätigkeiten Open Source Software vor allem für Kommunikations- und Testzwecke verwendet. Person 6 (Erstellung und Koordination der technischen Dokumentation), welche bereits seit 17 Jahren im Bereich der Softwaredokumentation tätig ist, gibt an, Open Source Software für die Erstellung der Dokumentation des proprietären Softwareprodukts des Unternehmens zu verwenden. Die dabei verwendeten Tools werden aber weder weiterentwickelt, noch in das proprietäre Softwareprodukt integriert.

Person 4 (Leitung der Produktentwicklungsabteilung) gibt in Bezug auf ihre Rolle und die Abteilung, für die sie verantwortlich ist, an, dass Produktion alles beinhaltet, was mit der Entwicklung des Produktes selbst und dessen Resultat zu tun hat. Da sich das proprietäre Softwareprodukt sehr stark aus Open Source Softwarekomponenten zusammensetzt, ist man als Produktionsleiterin bzw. Produktionsleiter dafür verantwortlich, dass bei der Produktion keine Fehler gemacht werden, die eventuell im Nachhinein viel Geld kosten können. Dazu gehören auch rechtliche Fehler.

Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) gibt in Bezug auf ihre Rolle an, technologische Trends und neue Möglichkeiten zu evaluieren um diese gegebenenfalls in das Produkt zu integrieren, um es technologisch relevant zu halten. Ihr Team beschäftigt sich mit dem Framework des proprietären Standardproduktes, also mit Technologien, die dem Produkt zugrunde liegen und das ganze Produkt durchziehen. In diesem Framework werden die Haupt-Open Source Komponenten definiert und deren Upgrades geplant. Die eingebundenen Open Source Softwareprodukte stehen dabei hauptsächlich unter Lizenzen ohne Copyleft, aber auch die LGPL ist in der Auflistung der Lizenzen vorhanden. Ebenso bestätigt Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) in Bezug auf ihre Rolle den beruflichen Zugang zur Thematik, indem sie sich als Softwarearchitektin bzw. Softwarearchitekt mit der Auswahl von Technologien und der Prüfung, ob die ausgewählten Produkte in das proprietäre Produkt integrierbar sind, beschäftigt.

In Bezug auf integrierte Open Source Softwareprodukte gibt Person 4 (Leitung der Produktentwicklungsabteilung) an, dass diese entweder direkt mitausgeliefert, oder dem Kunden als Voraussetzung für die Installation des proprietären Softwareprodukts vermittelt werden, für welche er selbst Sorge zu tragen hat. Dabei werden die entsprechenden Voraussetzungen im Installationshandbuch gelistet, sowie deren Abhängigkeiten.

Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) und Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) führen an, dass es innerhalb der Abteilung der Produktentwicklung zwei Eintrittswege für Open Source Software gibt, die es zu unterscheiden gilt. Einerseits sei dies die Planung künftiger Technologien, andererseits sei dies die Einbindung während der Entwicklung, um konkrete Problemstellungen zu lösen. Person 3

(Qualitätssicherung in der Produktentwicklungsabteilung) führt genauer aus, dass die zweite Variante dabei vordergründig auftritt. Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) ergänzt, dass ebenso Upgrades einen Eintrittsweg für neue Abhängigkeiten darstellen.

7.3.5 Lizenzmanagement

Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) gibt an, dass es an ihre Person zahlreiche Anfragen von verschiedenen Seiten, wie der des Projektmanagements, der Kunden, oder auch des Mutterkonzerns gibt, ob der Thematik des Lizenzmanagements und der damit verbundenen Thematik der Integration von Open Source Software in das proprietäre Softwareprodukt des Unternehmens, genügend Aufmerksamkeit geschenkt wird. Ihrer Aussage nach gibt es dafür keine eigene Rolle, die diese Thematik und die damit verbundenen Aufgaben abteilungsübergreifend verfolgt. Innerhalb der Abteilung der Produktintegration werde dieser Thematik jedenfalls noch sehr wenig Aufmerksamkeit geschenkt, man weiß aber zumindest von Initiativen der Produktentwicklungsabteilung, das Standardprodukt technisch mittels Code Analyse auf dessen Open Source Softwarebestandteile zu prüfen.

Darauf Bezug nehmend gibt Person 4 (Leitung der Produktentwicklungsabteilung) an, dass im Rahmen der Produktentwicklung die Aufgabe des Lizenzmanagements im Sinne von Auflistung aller verwendeten Open Source Softwarelizenzen im Rahmen des Releaseprozesses wahrgenommen wird. Ebenso sieht Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung) in dem Report, der das Ergebnis der technischen Analyse des Source Codes ist, kein aktives Lizenzmanagement. Jedenfalls bestätigen beide, dass der besagte Report in aufbereiteter Form Teil der Dokumentation, und somit Teil der Auslieferung jedes Releases ist. Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) führt näher aus, dass die Dokumentation aus dem Report manuell erfolgt, und eine Liste aller Abhängigkeiten, sowie deren Lizenz mit Lizenztext enthält. Ob die Art und Weise der Dokumentation den Bedingungen der Lizenzen entspricht, ist ihr nicht bekannt. Person 6 (Erstellung und Koordination der technischen Dokumentation) und 7 bestätigen ebenso, dass ein Report, indem die Lizenzen von allen Drittprodukten aufgelistet sind, Teil der Dokumentation ist. Dies ist in jedem Release Paket inkludiert und wird bei jedem Release aktualisiert. Wie die Prüfung erfolgt, ob die Art und Weise der Lizenzdokumentation den Bedingungen der Lizenzen entspricht, liegt im Aufgabenbereich von Person 4 (Leitung der Produktentwicklungsabteilung) bzw. der Rechtsabteilung und ist bislang nicht Aufgabe der Dokumentation. Person 4 (Leitung der Produktentwicklungsabteilung) bestätigt allerdings, dass die Art und Weise der Dokumentation bereits geprüft ist.

Person 6 (Erstellung und Koordination der technischen Dokumentation) ergänzt, dass es bezogen auf Tools zur Prozessunterstützung keine Anweisungen seitens des Unternehmens oder des Mutterkonzerns gibt, welche Tools heruntergeladen und eingesetzt werden dürfen. Es gibt auch keine gesammelte Liste, welche Tools firmenweit verwendet werden.

7.3.6 Prüfung und Freigabe

Sobald die Entscheidung getroffen wurde, neue Technologie einsetzen zu wollen, sieht Person 4 (Leitung der Produktentwicklungsabteilung) die Prüfung der rechtlichen Rahmenbedingungen als Aufgabe des Produktmanagements. Bei bestimmten Lizenzen kann generell auf Prüfung und Freigabe verzichtet werden, wenn bekannt ist, dass diese kompatibel mit den Absichten im Zusammenhang mit dem Geschäftsmodell hinter dem proprietären Softwareprodukt sind.

Die Ansammlung an Tätigkeiten in Bezug auf die Prüfung und Freigabe eines Releases, wird von Person 4 (Leitung der Produktentwicklungsabteilung) orchestriert. Diese erwähnt dabei, dass diese Tätigkeiten zwar bereits dokumentiert, aber noch nicht in ausreichender Form in einem Prozess dargestellt und kommuniziert sind. Außerdem ist die Durchführung der Tätigkeiten auf wenige Personen beschränkt.

In Bezug auf die Prüfung und Freigabe der Einbindung von Open Source Software in das Standardprodukt gibt Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung) an, dass dies innerhalb der Abteilung der Produktentwicklung auf der Ebene der persönlichen Kommunikation, und ohne Prozess geschieht. Dies bestätigt Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) als gelebte Praxis. Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) und 6 äußern, ebenso keinen klaren Prozess in Bezug auf die Prüfung und Freigabe von Open Source Software innerhalb des Unternehmens oder der Abteilung zu kennen. Ferner sind Details bezogen auf die technische Analyse des Source Codes, sowie genauere Informationen darüber, wie die Dokumentation von Open Source Software erfolgt, nicht allen Mitarbeiterinnen und Mitarbeitern bekannt.

In Zusammenhang mit der Produktdokumentation äußert Person 6 (Erstellung und Koordination der technischen Dokumentation), dass ihr nicht bekannt sei, ob die Art und Weise der Dokumentation der vorhandenen Lizenzen bereits von der Rechtsabteilung geprüft wurde und den Lizenzbedingungen der angeführten Lizenzen entspricht.

Der Entscheidungsprozess, wie es zu eingebundenen Open Source Softwareprodukten kommt, wird von Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung) als intransparent wahrgenommen. Weiter bemängelt diese, dass das Qualitätsmanagement bislang zu wenig in die Thematik der Open Source Softwareintegration eingebunden ist. Dieses kann daher weder Alternativen prüfen, noch welche Rahmenbedingungen bei der Auswahl berücksichtig wurden.

Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) und 6 erwähnen beide, dass sie bei Anfragen an die Rechtsabteilung zu Open Source Themen keine Antwort erhalten haben, wodurch ihnen nicht bekannt ist, an wen sie welche Anfragen richten können und sollen, und wer für die Beantwortung der Fragen zuständig ist.

Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) und 7 merken des Weiteren an, dass ihnen nicht bewusst ist, wie andere Abteilungen des Unternehmens mit dieser Thematik umgehen.

7.3.7 Maßnahmen zur Risikovermeidung

In Bezug auf die Risikovermeidung innerhalb der Abteilung der Produktintegration gibt Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) an, dass eine Prüfung auf enthaltene Lizenzen und der Erfüllung derer Lizenzbedingungen nur automatisiert möglich ist, und nur von entsprechend geschultem Personal durchgeführt werden kann, um die Ergebnisse richtig interpretieren zu können. Innerhalb der Abteilung der Produktintegration gibt es keine derartige Prüfung, auch gibt es keinen Prozess, der auf den richtigen Umgang mit der Integration von Open Source Softwareprodukten abzielt.

In Bezug auf die Schulung von Mitarbeitern wurden laut Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung), 3 und 4 bislang zwei Termine veranstaltet, in welchenGrundlagen zum Thema Open Source Software durch die Rechtsabteilung vermittelt wurden. Dies kann als erster Schritt für eine Trainingsinitiative verstanden werden. Es fehlt aber an klar dokumentierten und kommunizierten Prozessen, wie mit dieser Thematik in der täglichen Arbeit umgegangen werden soll, wodurch die Mitarbeiterinnen und Mitarbeiter der Abteilung nur nach bestem Wissen und Gewissen handeln können.

In Bezug auf die Auswahl von Open Source Softwareprodukten werden laut Person 4 (Leitung der Produktentwicklungsabteilung) Firmen bevorzugt, die sich mit der rechtlichen Thematik hinter dem Open Source Softwaremodell beschäftigen, klare Lizenzmodelle und Lizenzbedingungen transportieren und ihre Softwareprodukte als Open Source Softwareprodukte anbieten.

Hinsichtlich der Prüfung des Standardproduktes geben Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung), 4 und 7 an, dass mittels eines Tools über Nacht der Source Code des proprietären Softwareprodukts gescannt, und auf vorhandene Abhängigkeiten, demzufolge auch auf Abhängigkeiten zu Open Source Softwareprodukten, analysiert wird. Dieses Tool kann ebenso feststellen, ob sogenannte Source Codes Snippets (einzeln kopierte Source Code Teile) von Open Source Softwareprodukten stammen. Für die Auswertung dieser technischen Analyse sind Experten des Mutterkonzerns zuständig, welche auf den Umgang mit diesem Tool und der Interpretation der Ergebnisse spezialisiert sind.

7.4 Erhebung des SOLL-Zustandes

Aufgrund der durchgeführten Expertinnen- und Expertenbefragung, konnte folgender Soll-Zustand erhoben werden, welcher in die Definition des Open Source Genehmigungsprozesses einfließt:

7.4.1 Prozessdefinition und Leitfaden

Person 6 (Erstellung und Koordination der technischen Dokumentation) nennt in Bezug auf den Soll-Zustand, dass man selbst sicherstellen können muss, alle nötigen Aspekte bedacht zu haben. Um dies gewährleisten zu können, müssen Information über Prozessschritte und Abläufe allen Mitarbeitern kommuniziert und zugänglich sein, damit diese sich informieren und

entsprechend handeln können. Des Weiteren sollen die Tätigkeiten in Zusammenhang mit Open Source und deren Ausführung auf die unterschiedlichen Rollen verteilt werden, damit diese nicht auf wenige Personen konzentriert sind.

In Bezug auf die Einführung eines Open Source Genehmigungsprozesses bedarf es einer übergreifenden Rolle, die für den richtigen Umgang mit Open Source Software verantwortlich ist, da alle Abteilungen die gleichen Regeln anwenden müssen, so Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung). Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) führt näher aus, dass es in der Verantwortung dieser Rolle liegt, einen definierten Prozess zu leben und zu betreuen. Sie soll als verlängerter Arm der Rechtsabteilung verstanden werden, welcher vor Ort prozessverantwortlich und ebenso durchführungsverantwortlich ist. Diese soll ebenso verantwortlich für Freigabe der einzubindenden Open Source Softwareprodukten sein. Dabei ist zu unterscheiden, ob es sich um Anfragen handelt, die während der Entwicklung auftauchen, oder um größere Themengebiete, die im Vorhinein im Rahmen des Produkt Managements geplant werden. Jedenfalls soll die Entwicklung durch den Prozess nicht aufgehalten werden. Dem stimmt ebenso Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) zu, da ansonsten entweder der Prozess nicht beachtet, oder aufgehört wird, Open Source Softwareprodukte zu verwenden.

In Bezug auf die bestehenden Rollen innerhalb des Unternehmens haben Softwareentwicklerinnen und Softwareentwickler darauf zu achten, dass sie das, was sie tun möchten, auch dürfen. Der Produktmanager hat die Aufgabe, sicherzustellen, dass für den Teil der Softwareentwicklung, den er zu verantworten hat, die Prozesse rund um den gewünschten Einsatz von Open Source Software, eingehalten werden. Der Tester kann unterstützen, wenn er entsprechend geschult ist. Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung), 3 und 7 sind sich einig, dass alle vorhandenen Rollen in Bezug auf einen Open Source Genehmigungsprozess betroffen sind, wenn dieser auf den gesamten Produktentwicklungsprozess angewendet wird. Daher müssen auch alle Rollen über die gesamte Thematik ausreichend Bescheid wissen.

In Bezug auf den Prozess merkt Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) an, dass dieser aufzeigen muss welche Schritte notwendig werden, wenn die technische Prüfung einen problematischen Umgang mit der Einbindung von Open Source Software in Hinblick auf das Geschäftsmodell des Unternehmens aufdeckt. Dabei ist das Fehlverhalten von Entwicklern zu behandeln sowie die Auswirkungen auf das proprietäre Softwareprodukt. Dem pflichtet Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung) bei, die anführt, dass die Ergebnisse der technischen Analyse den Architekten übermittelt werden soll, damit diese den richtigen Umgang mit Open Source Software innerhalb ihrer Softwareentwicklungsteams sicherstellen können.

In Bezug auf die Erstellung eines Open Source Genehmigungsprozesses gibt Person 4 (Leitung der Produktentwicklungsabteilung) an, dass es eine klare Dokumentation darüber geben muss, wie neue Technologien in das proprietäre Softwareprodukt des Unternehmens eintreten können, inklusive Entscheidungsgremium. Demzufolge ist der Produktentwicklungsprozess in Bezug auf

die Open Source Thematik zu dokumentieren, bei welchem ebenso verantwortliche Rollen, deren Aufgaben, sowie zu übermittelnde Artefakte erfasst werden müssen.

Aus Sicht des Qualitätsmanagements ist es laut Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung) notwendig, die Evaluierung von potentiellen Open Source Softwareprodukten in den Prozess aufzunehmen, und einen Kriterienkatalog zu definieren, der für die Prüfung relevant ist. Ebenso sind getroffene Entscheidungen laut Person 6 (Erstellung und Koordination der technischen Dokumentation) nachvollziehbar zu begründen und dokumentieren

In Bezug auf das betreffende Open Source Softwareprodukt betont Person 5 (Technologie-Innovation in der Produktentwicklungsabteilung), dass eine Abstimmung mit den Softwarearchitekten bei der Einführung aller Open Source Komponenten notwendig ist, auch wenn diese aus rechtlicher Sicht kein Problem darstellen. Damit soll verhindert werden, dass etwa mehrere Tools zur Erfüllung desselben Zwecks eingesetzt werden. Ansonsten kann ein Wildwuchs an Abhängigkeiten nicht verhindert werden. Außerdem kann dies zu Sicherheitsproblemen führen, da beispielsweise in einer älteren Version einer Softwarebibliothek ein Sicherheitsproblem besteht, dieses aber in einer neueren Version bereits behoben ist. Lediglich die Prüfung des Source Codes mittels des Tools und die rechtliche Unbedenklichkeit sind demzufolge nicht ausreichend.

In Bezug auf den zu erstellenden Leitfaden gibt Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) an, dass dieser als Kurzanleitung dienen soll, bei welchem auf einer A4 Seite Informationen über verschiedene Open Source Lizenzen, der jeweilige Handlungsbedarf, sowie ein Link zum Open Source Genehmigungsprozess gelistet sind. Dem schließt sich Person 4 (Leitung der Produktentwicklungsabteilung) an, die den Leitfaden als Regelwerk für Softwareentwicklerinnen und Softwareentwickler sieht, die nicht lange auf Antworten durch die Rechtsabteilung warten können. Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) konkretisiert, dass die jeweiligen Ansprechpersonen mit Kontaktinformation zu listen sind, ebenso wie die zu erwartenden Antwortzeiten. Sowohl positive als auch negative Entscheidungen sollen innerhalb eines Tages möglich sein, um die Entwicklung nicht zu verzögern, so Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung).

7.4.2 Folgeschritte

Beim Ausrollen des Open Source Genehmigungsprozesses ist laut Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) und 6 darauf zu achten, dass der Prozess flächendeckend bekannt gemacht wird. Sie schlagen dabei den Top-Down Ansatz vor, wobei Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) genauer ausführt, zuerst Produktmanager, dann Scrum Master und Senior Entwicklerinnen und Senior Entwickler, und anschließend die Entwicklungsteams zusammen mit den bereits genannten Rollen über den Prozess zu informieren und zu schulen, um für ein einheitliches Verständnis sorgen zu können. Person 3 (Qualitätssicherung in der Produktentwicklungsabteilung) und 6 merken an, dass es dabei wichtig ist, zu erklären, warum diese Thematik von Relevanz ist, um Bewusstseinsbildung bei den Mitarbeiterinnen und Mitarbeitern zu schaffen. Aufgrund des Risikos für das

Unternehmen, schlägt Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) ebenso vor, dass bei Fehlverhalten, Konsequenzen für die betroffenen Mitarbeiterinnen und Mitarbeiter zu ziehen sind.

Ferner schlagen Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration) und 6 vor, die Dokumentation des neuen Prozesses an die Art und Weise der bereits vorhandenen Prozesse anzugleichen, um die Akzeptanz zu erhöhen. Ebenso sei die Dokumentation des proprietären Softwareprodukts gegen die Bedingungen der eingebundenen Open Source Software Lizenzen zu prüfen.

In Bezug auf die Schulung von Mitarbeiterinnen und Mitarbeitern schlägt Person 2 (Produktmanagement innerhalb der Produktentwicklungsabteilung) vor, in der bestehenden Basisschulung auf den erarbeiteten Prozess zu verweisen. Darüber hinaus soll die Thematik in das jährlich verpflichtende Online-Security-Training des Konzerns aufgenommen werden, sowie ein eigenes Online-Training, welches nur auf den Umgang mit Open Source Software abzielt, bedacht werden, so Person 1 (Qualitätssicherung und Prozessmanagement Produktintegration). Außerdem soll laut Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) der definierte Prozess sowie der Leitfaden in die Link-Sammlung für neue Mitarbeiter aufgenommen werden, da diese Artefakte von Beginn an wichtige Informationen darstellen.

Bezogen auf die Ausrichtung des Unternehmens ist Person 7 (Softwarearchitektur in der Produktentwicklungsabteilung) der Meinung, dass man mit Eigenentwicklungen mehr nach außen gehen und unter Open Source stellen soll, da man dadurch positive Reputation innerhalb der Community erhält und nicht nur Nutznießer ist.

7.5 Fazit

Die befragten Expertinnen und Experten konnten die Vor- und Nachteile von Open Source Softwareprodukten, die im theoretischen Teil erarbeitet wurde, bestätigen, und mit praxisbezogenem Hintergrundwissen näher erläutern.

Innerhalb des Unternehmens, in dem die befragen Expertinnen und Experten tätig sind, kann die IST-Situation in Bezug auf die Integration von Open Source Softwareprodukten in das kommerzielle, proprietäre Softwareprodukt, welches im Rahmen des Unternehmens hergestellt und vertrieben wird, wie folgt zusammengefasst werden:

- Technische Source Code Analyse wird bereits durchgeführt
- Abhängigkeiten zu andern (Open Source) Softwareprodukten werden dokumentiert
- Detailwissen und Tätigkeiten sind auf wenige Personen verteilt
- Entscheidungsfindung wird als intransparent wahrgenommen
- Abteilungsübergreifend wird unterschiedlich vorgegangen
- Durchgeführte Basisschulung wird als unzureichend empfunden

- Zuständigkeitsbereiche und Ansprechpersonen der Rechtsabteilung sind nicht ausreichend bekannt
- Der Prozessschritt der Überprüfung der Dokumentation gegen die Lizenzbedingungen ist vielen Expertinnen und Experten nicht bekannt
- Es gibt keine dokumentierten und kommunizierten Prozesse

Daraus ergibt sich aus Sicht der Expertinnen und Experten die Notwendigkeit eines Prozesses, der folgende Punkte berücksichtigt, um den SOLL-Zustand zu erreichen:

- Definition und Visualisierung aller Abläufe
- Festlegung von Verantwortlichkeiten
- Verteilung der gebündelten Tätigkeiten auf mehrere Rollen
- Einheitliche und nachvollziehbare Evaluierung
- Rasche Antwortzeiten, um Entwicklung nicht aufzuhalten
- Definition von Ansprechpartnern
- Ablage des Prozesses an einer für alle Mitarbeiterinnen und Mitarbeiter zugänglichen Stelle

Ferner ergeben sich Folgeschritte, die im Anschluss an die Definition des Prozesses zu berücksichtigen sind:

- Information und Schulung über den Prozess
- Integration des vorhandenen Ticketsystems
- Ausweitung des Schulungsmaterials
- Einheitliche Vorgehensweise im Unternehmen

Diese gesammelten und aufbereiteten Informationen bilden die Ausgangsbasis für die Erstellung eines Open Source Genehmigungsprozesses sowie eines Leitfadens, welche im folgenden Kapitel näher erläutert werden.

8 DEFINITION DES OPEN SOURCE GENEHMIGUNGSPROZESSES

Durch die theoretische Aufarbeitung des Themengebiets und die darauffolgende Befragung von Expertinnen und Experten, konnte ein erster Entwurf eines Open Source Genehmigungsprozesses erarbeitet werden, welcher anschließend mit den befragten Expertinnen und Experten begutachtet wurde. Nach Einarbeitung des Feedbacks konnte der Genehmigungsprozess festgelegt, und dessen Umsetzung geplant werden. Gleichzeitig wurde ein Leitfaden für den Umgang mit Open Source Softwareprodukten erarbeitet, der dieselben Phasen der Erstellung durchlief.

Bei der Erarbeitung des Open Source Genehmigungsprozesses wurde der Kernprozess des Produkt Managements berücksichtigt, in dem das kommerzielle, proprietäre Softwareprodukt des Unternehmens hergestellt wird, da es diesen im Rahmen dieser Arbeit anzupassen galt.

Die Anpassung weiterer Prozesse wird in den Folgeschritten in Kapitel 9 gelistet, aber nicht näher hinsichtlich der genauen Anpassung beschrieben.

8.1 Festlegung des Open Source Genehmigungsprozesses

Folgender Open Source Genehmigungsprozess konnte im Anschluss an die Interviews und Reviewgespräche mit den Expertinnen und Experten definiert werden, in welchem alle Expertinnen und Experten die Interessen ihrer jeweiligen Zuständigkeitsbereiche vertreten sahen. Abschließend konnte der definierte Prozess an die Rechtsabteilung des Konzerns übergeben werden.

Die Beschreibung des Prozesses gliedert sich wie folgt:

- Kontextuelle Einordnung – Prozesshaus des Unternehmens
- Begriffsdefinitionen
- Der Produkt Management Prozess
- Anwendung des Open Source Genehmigungsprozesses
- Beschreibung der Rollen

Die Inhalte wurden dabei in englischer Sprache verfasst, um den Rahmenbedingungen des Unternehmens zu entsprechen.

8.1.1 Kontextuelle Einordnung – Prozesshaus des Unternehmens

Anhand des erarbeiteten theoretischen Hintergrundes sowie der durchgeführten Interviews mit Expertinnen und Experten sind in Anlehnung an (Ruffing, 2017) folgende Prozesse anzupassen bzw. zu entwickeln:

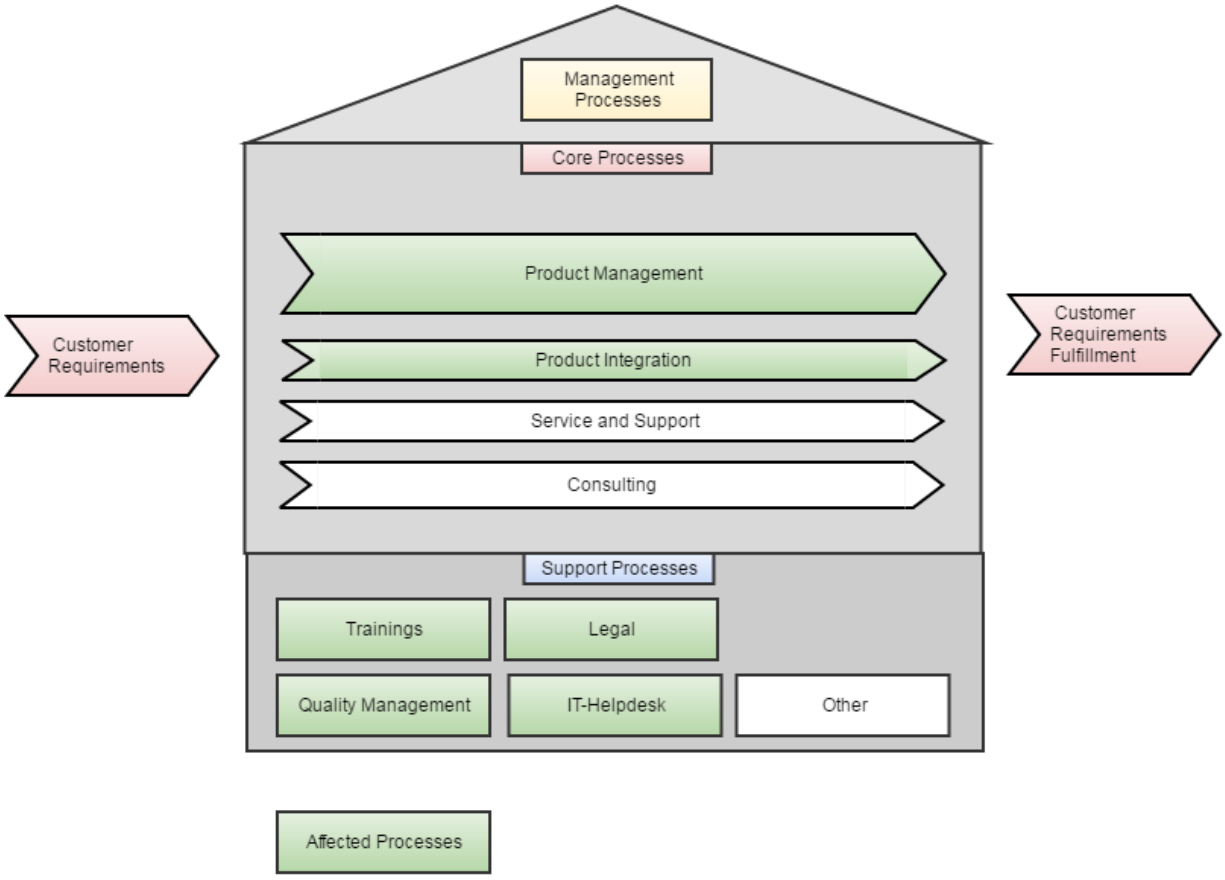

Abbildung 9 - Vereinfachte Darstellung des Prozesshauses des Unternehmens

Bei dem Produkt Management Prozess handelt es sich um den Prozess, in welchem Rahmen das kommerzielle, proprietäre Softwareprodukt des Unternehmens entwickelt wird. Dieser soll von dem Open Source Genehmigungsprozess beeinflusst werden. Der Open Source Genehmigungsprozess selbst fällt unter den Supportprozess „Legal“, und ist in diesen einzugliedern. Der IT-Helpdesk ist insofern betroffen, als dieser eine neue Ticketkategorie für rechtliche Belange, wie die Anfragen hinsichtlich des Einsatzes von Open Source Softwareprodukten, zur Verfügung stellen soll. Der Supportprozess des Qualitätsmanagements, welcher unternehmensweit Qualitätssicherungsstrategien und deren Umsetzung verfolgt, soll die Thematik des Einsatzes von Open Source Software im kommerziellen, proprietären Softwareprodukt des Unternehmens aufgreifen und in die entwickelten Strategien und deren Umsetzung integrieren. Der Supportprozess des Trainings, welcher die Weiterbildungsmaßnahmen der Mitarbeiterinnen und Mitarbeiter beinhaltet, soll ebenfalls um die Thematik des Einsatzes von Open Source Software in proprietären Softwareprodukten erweitert werden. Ebenso ist der Open Source Genehmigungsprozess in den Kernprozess der Produktingegration zu integrieren, da für die Anpassung des proprietären Softwareprodukts auf kundenspezifische Bedürfnisse dieselben Kriterien von Relevanz sind. Die Anpassung dieses Kernprozesses erfolgt, ebenso wie die Anpassung der Supportprozesse des Trainings und des Qualitätsmanagements, nach erfolgreicher Umsetzung im Kernprozess des Produkt Managements. Diese Anpassungen werden daher in den Folgeschritten in Kapitel 9 beschrieben.

Wie erläutert, werden folgende Elemente des Prozesshauses behandelt:

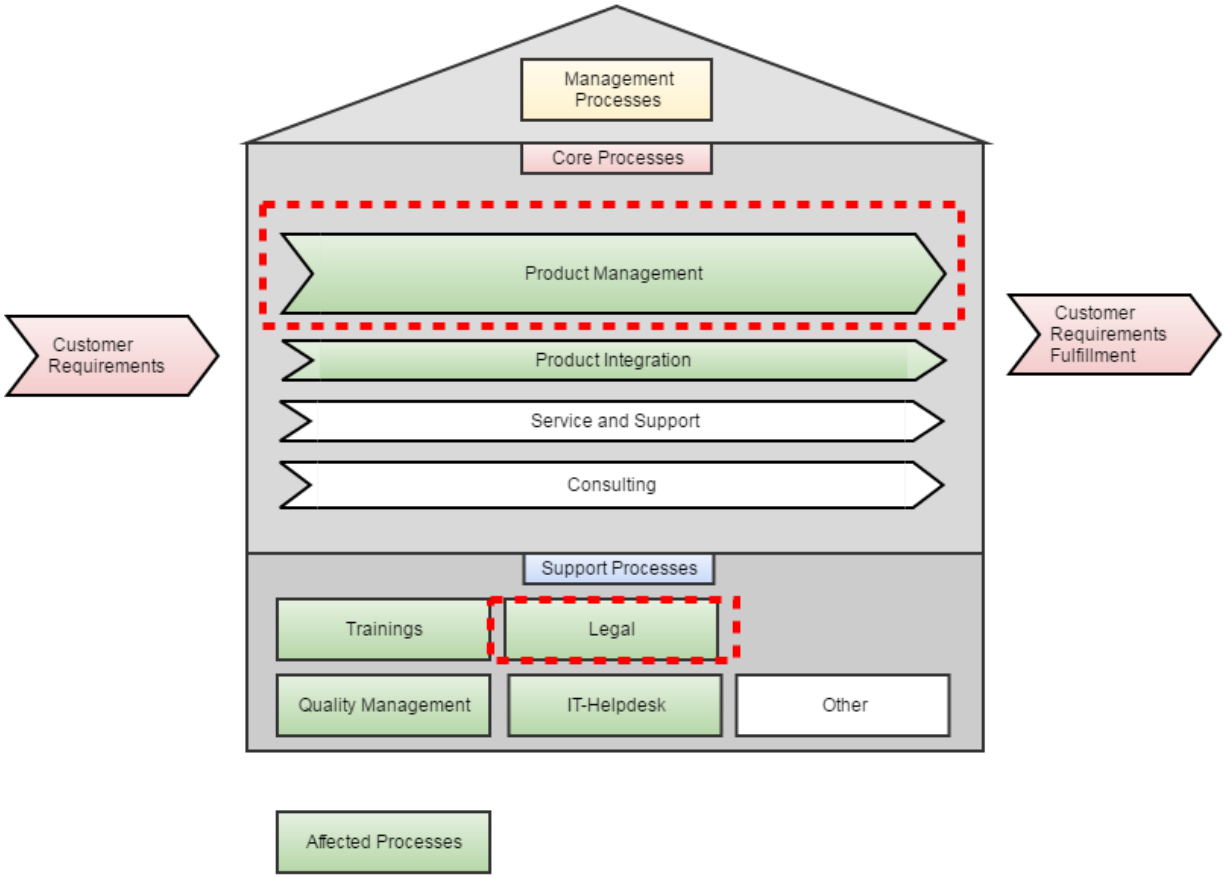

Abbildung 10 - Ausgewählte Prozesse des Prozesshauses

In folgenden Abschnitten werden der Produkt Management Prozess, der Open Source Genehmigungsprozess, welcher dem Kontext des Legal-Supportprozesses zuzuordnen ist, sowie deren Begrifflichkeiten, Interaktionen und Rollen erläutert.

8.1.2 Begriffsdefinitionen

Da die Begriffsdefinitionen Teil des dokumentierten Prozesses sind, wurden diese ebenso in englischer Sprache verfasst.

Open Source Approval Process

- The open source approval process is a process that is introduced to accompany the treatment rules related to the integration of open source code into the source code of the proprietary software product.

Product Management Process

- The product management process shows the end-to-end process of developing the product and includes following steps: scoping, design, development, release test, release delivery and idea generation process.

Idea Generation Process

- The idea generation process is part of the scoping phase of the product management process and describes the process steps of potentially new functionality of the product which has to be consolidated, evaluated, documented and prioritized. Afterwards the solution outline of the new functionality has to be introduced to the idea generator and other stakeholders. The new functionality is introduced to the roadmap depending on its priority against other new functionality which will be developed.

Solution Outline

- The solution outline is a deliverable which is created in the scoping phase of the product management process and describes new functionality of a feature in detail that will be implemented in the product. It lists the vision behind the feature, the business value(s), the way of realization, non-functional requirements, limitations and out of scopes.

Abbildung 11 - Begriffsdefinitionen des Open Source Genehmigungsprozesses Teil 1

Abbildung 12 - Begriffsdefinitionen des Open Source Genehmigungsprozesses Teil 2

8.1.3 Der Produkt Management Prozess

Als Ausgangspunkt für die Beschreibung und Anwendung eines Open Source Genehmigungsprozesses wurde der Produkt Management Prozess visualisiert, und potentielle Interaktionspunkte mit dem Open Source Genehmigungsprozess definiert.

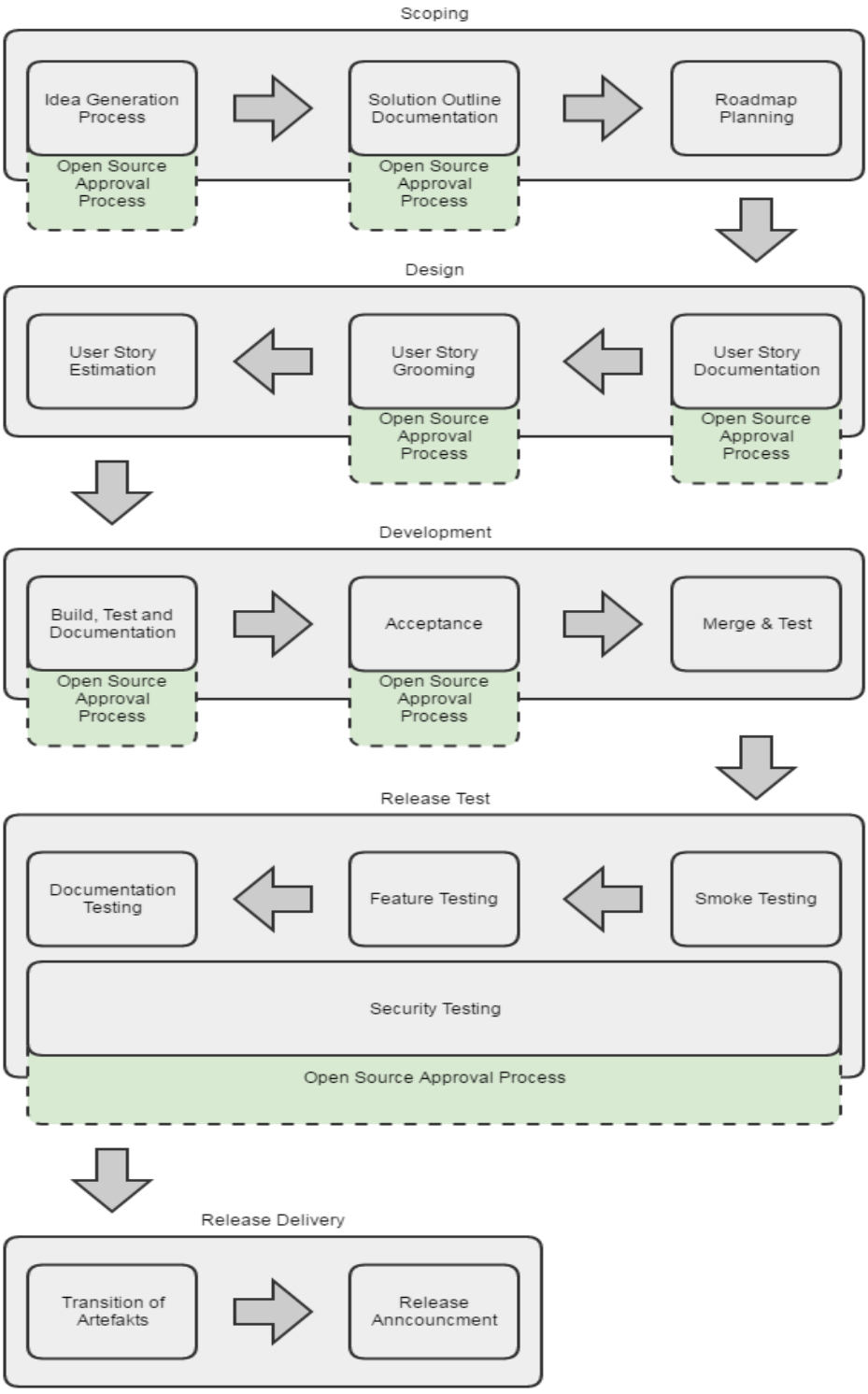

Abbildung 13 - Der Produkt Management Prozess

8.1.4 Anwendung des Open Source Genehmigungsprozesses

In weiterer Folge konnte der Open Source Genehmigungsprozess definiert werden, wobei sich herausstellte, dass dieser abhängig von der Phase des Produkt Management Prozesses, unterschiedliche Ausprägungen und involvierte Rollen aufweist.

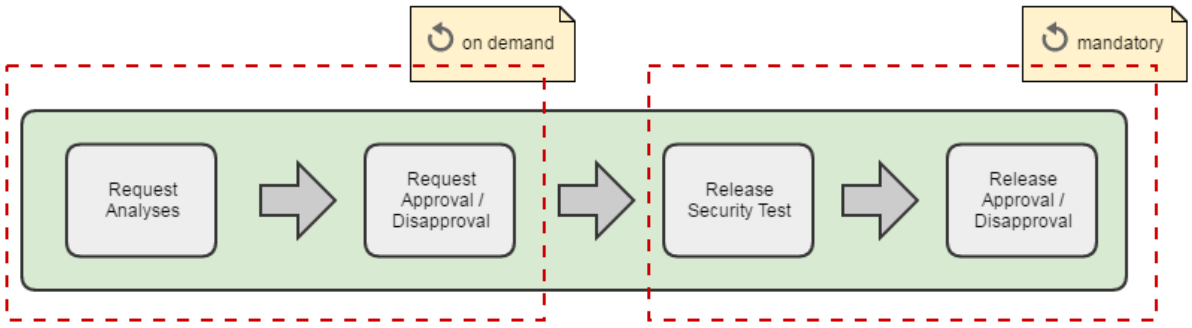

Abbildung 14 - High-Level Open Source Genehmigungsprozess

Im ersten Teil des Open Source Genehmigungsprozesses handelt es sich um Prozessschritte, die nur nach Bedarf, also dann, wenn ein Open Source Softwareprodukt eingebunden werden soll, durchzuführen sind. Im zweiten Teil handelt es sich um die Prozessschritte, die verpflichtend am Ende jedes Releases des proprietären Softwareprodukts durchzuführen sind.
Abgebildet auf den Produkt Management Prozess sieht dies wie folgt aus:

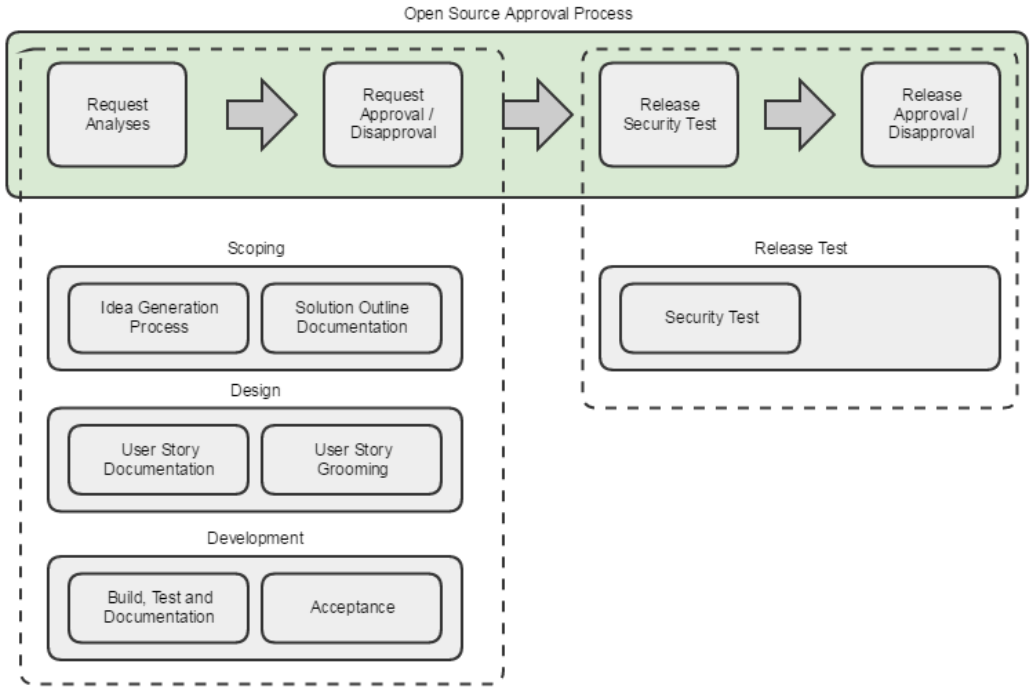

Abbildung 15 - Open Source Genehmigungsprozess Mapping

In Abbildung 15 wird ersichtlich, dass in den Phasen des Scopings, Designs und Developments der erste Teil des Open Source Genehmigungsprozesses Anwendung findet, und in der Phase

des Release Tests der zweite Teil des Open Source Genehmigungsprozesses verpflichtend durchzuführen ist.

Der erste Teil des Open Source Genehmigungsprozesses stellt sich wie folgt dar:

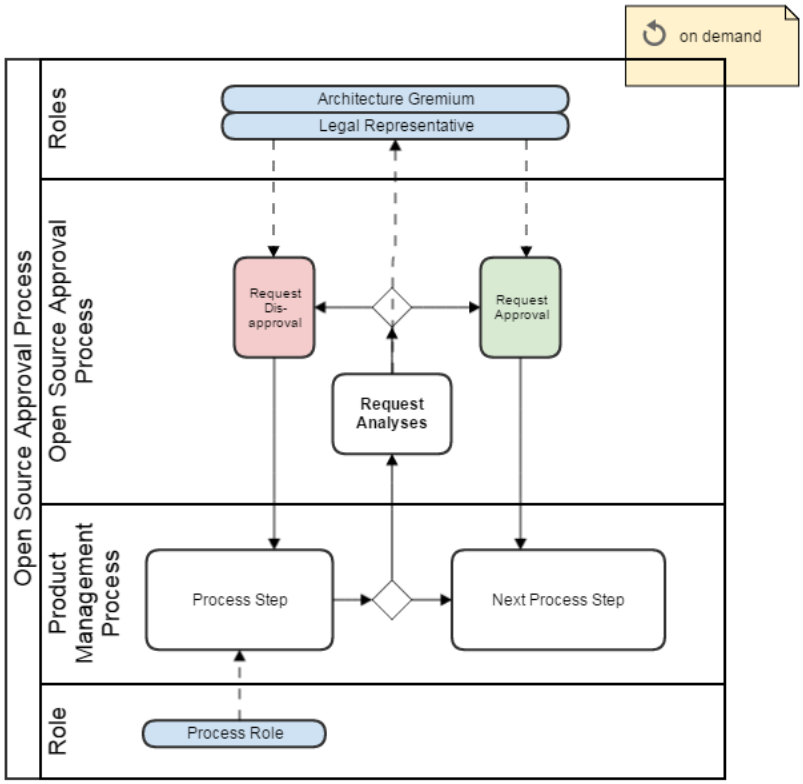

Abbildung 16 - Open Source Genehmigungsprozess 1. Teil

Bei Bedarf stellt die jeweilige verantwortliche Rolle des betreffenden Prozessschrittes des Produkt Management Prozesses eine Anfrage bezüglich der Einbindung des betroffenen Open Source Softwareprodukts. Folglich prüft diese, wie detailliert in Abbildung 18 zu sehen ist, mit Hilfe des Leitfadens, ob neben der Abstimmung mit dem Architekten Gremium auch eine separate Anfrage an die Rechtsabteilung zu stellen ist. Jedenfalls ist ein Termin mit dem Architekten Gremium zu vereinbaren, in welchem anhand einer standardisierten Evaluation entschieden wird, ob man das entsprechende Open Source Softwareprodukt in das proprietäre Softwareprodukt des Unternehmens integrieren möchte. Die Evaluation sowie die Entscheidung bilden ein Artefakt, welches aus diesem Termin hervorgeht und an allgemein zugänglicher Stelle abgelegt wird. Kann anhand der Prüfung mittels des Leitfadens auf die Analyse der Rechtsabteilung verzichtet werden, ist die Entscheidung des Architekten Gremiums ausschlaggebend. Ansonsten muss die Entscheidung der Rechtsabteilung abgewartet werden. In diesem Fall müssen sich beide Analysen für den Einsatz des entsprechenden Open Source Softwareprodukts aussprechen, um es tatsächlich einbinden zu dürfen. Wird die Anfrage abgelehnt, verbleibt man im ursprünglichen Prozessschritt und muss nach Alternativen suchen. Wird der Anfrage zugestimmt, kann in den nächsten Prozessschritt übergegangen werden. Gibt es im jeweiligen

Prozessschritt keine Anfrage zur Einbindung von Open Source Softwareprodukten, findet auch der Open Source Genehmigungsprozess keine Anwendung.
Der zweite Teil des Open Source Genehmigungsprozesses stellt sich wie folgt dar:

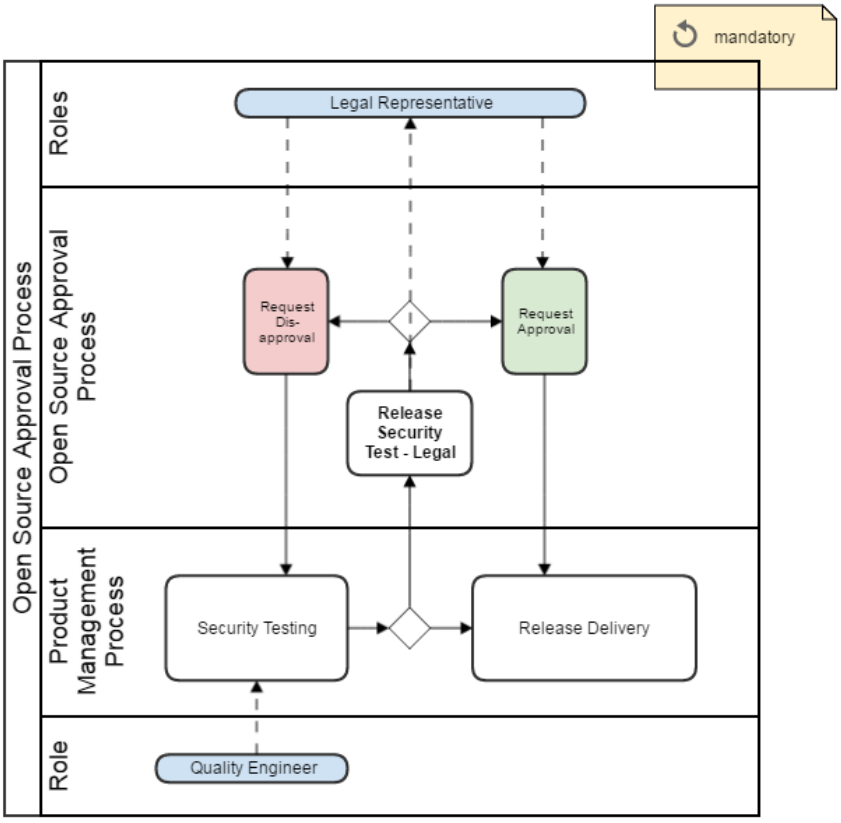

Abbildung 17 - Open Source Genehmigungsprozess 2. Teil

Der Qualitätsingenieur stellt wie in Abbildung 19 detailliert dargestellt wird, in Zusammenarbeit mit den anderen Rollen sicher, dass alle Artefakte des Releases fertig gestellt sind. Anschließend startet dieser die automatisierte Source Code Analyse. Darauf folgend überprüft die Abteilung der Dokumentation den Report der technischen Analyse erneut auf mögliche Änderungen, und pflegt diese in den entsprechenden Teil der Lizenzabhängigkeiten ein. Der Report, sowie die gesamte Dokumentation werden anschließend an die Rechtsabteilung übermittelt, welche nun dafür verantwortlich ist, das Release und die zugehörige Dokumentation freizugeben. Der Qualitätsingenieur wird darauf folgend schriftlich von der Rechtsabteilung über die Freigabe bzw. die begründete Ablehnung der Freigabe des Releases verständigt. Die Darstellung des Open Source Genehmigungsprozesses in Anwendung auf den gesamten Produkt Management Prozess ist in Abbildung 20 zu sehen, wobei sich hier die involvierten Rollen in den verschiedenen Phasen unterscheiden. Explizit zu erwähnen ist, dass jede Ablehnung einer Open Source bezogenen Anfrage in einem Prozessschritt nach der Phase „Build, Test and Documentation“ dazu führt, dass der Produkt Management Prozess wieder in diese Phase zurückfällt, da nur in dieser die nötigen Änderungen am Source Code durchgeführt werden können.

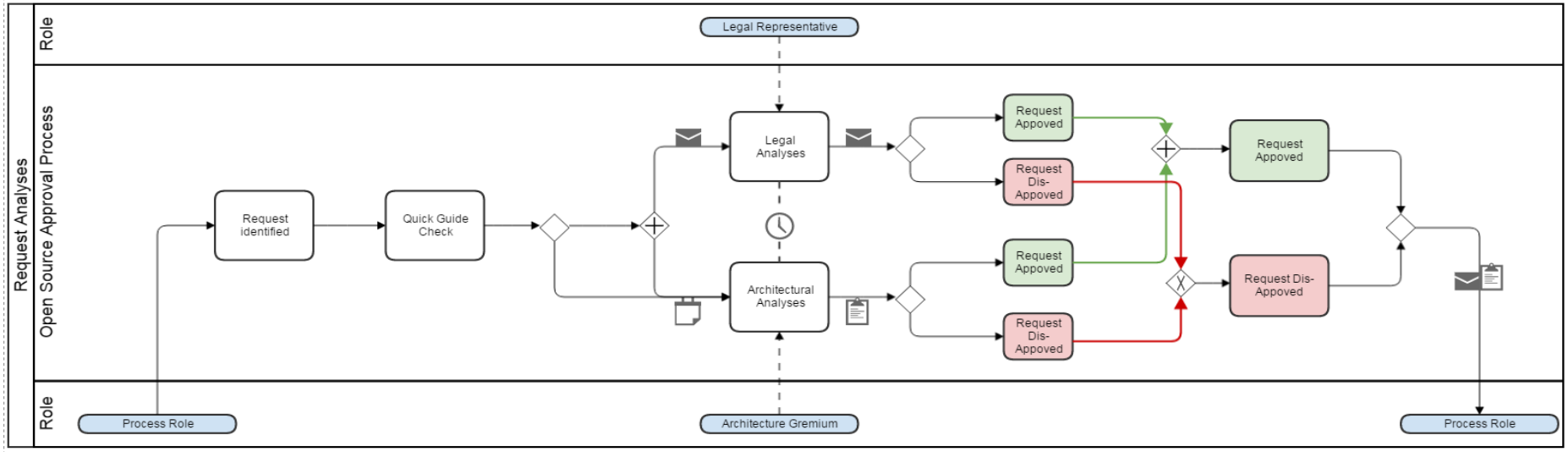

Abbildung 18 - Detaillierter Open Source Genehmigungsprozess 1. Teil

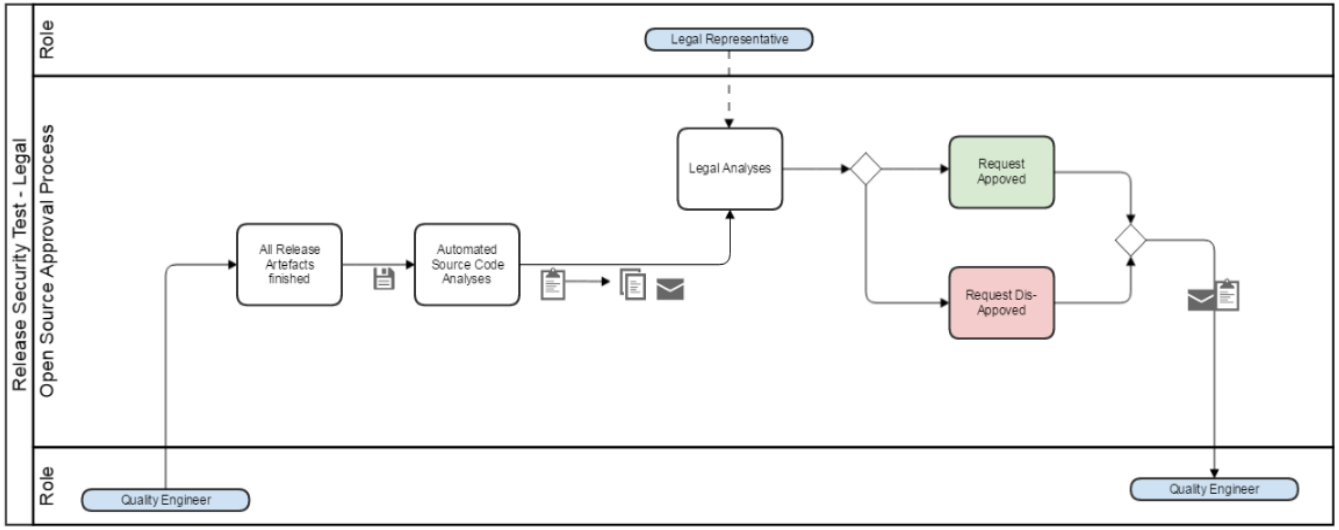

Abbildung 19 - Detaillierter Open Source Genehmigungsprozess 2. Teil

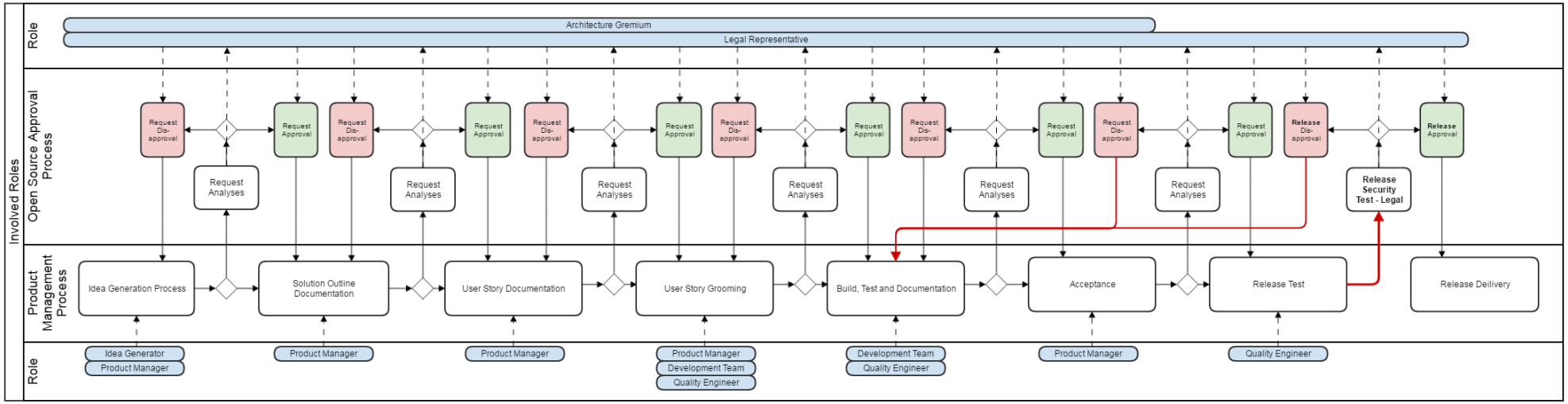

Abbildung 20 - Der Open Source Genehmigungsprozesses im Produkt Management Prozess

8.1.5 Beschreibung der Rollen

Wie folgt werden die betroffenen Rollen beschrieben:

Idea Generator

- An idea generator is someone who raises new ideas for functionality that should be brought into the software product. This idea generator can be any person or role inside and outside the company.

Product Manager

- A product manager is someone who is part of product management team and owns certain features of the product from an end to end perspective. The product manager can also be defined as product owner. In the phase of the idea generation process the product manager works together with the idea generator to check the intention of the idea against the vision of the product. Depending on the size and the socpe, an idea can either lead to one or more featuers, or a feature could consist of one or more ideas. In the phase of the solution outline documentation a product manager is resposible for defining the solution outline document, which describes a new feature which will be developed within an upcoming release. Within the phase of the user story documentation a product manager is accountable for documenting the user stories of a feature. In the phase of the user story grooming a product manager assures that the user stories on top of the product backlog are understood by the development team. In the phase of the acceptance a product manager verifys, that an user story was realized according to the documentation.

Development Team

- A development team is a group of software engineers, who develop new features of the software product. Within the phase of the user story grooming the development team is responisble for asking all questions that are necessary to understand the intention and the size of a user story. Within the phase of build, test and documentaion the development team develops and tests user stories according to the user story documentation and the development guidelines of the company. Also the documentation of the product itself has to be updated according to the newly developed functionality.

Abbildung 21 - Rollenbeschreibungen des Open Source Genehmigungsprozesses Teil 1

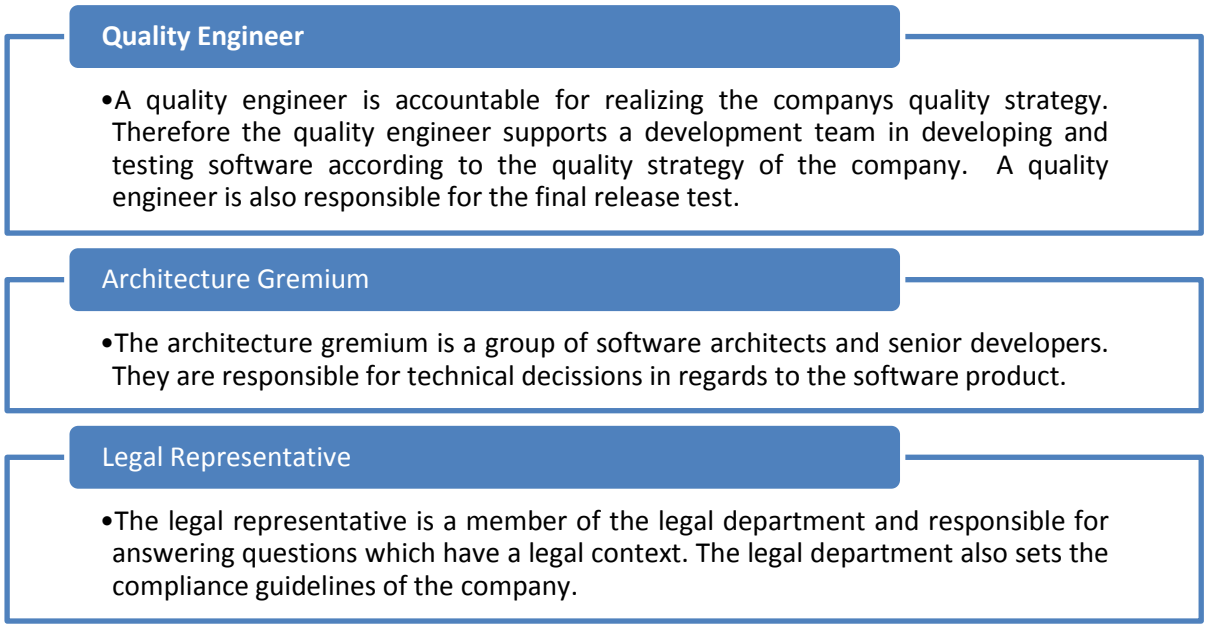

Abbildung 22 - Rollenbeschreibungen des Open Source Genehmigungsprozesses Teil 2

8.2 Ableitung eines generischen Open Source Genehmigungsprozesses

Um einen generischen Open Source Genehmigungsprozess ableiten zu können, müssen die firmenspezifischen Elemente des Open Source Genehmigungsprozesses entfernt und durch die Prozesse, Phasen und Rollen ersetzt werden, auf die der Open Source Genehmigungsprozess angewandt werden soll. In Scrum orientierten Softwareentwicklungsunternehmen kann der beschriebene Prozess in bestehender Form als Vorlage verstanden werden, da innerhalb des Unternehmens, in dem der Open Source Genehmigungsprozess definiert wurde, ebenso mit Scrum gearbeitet wird.

8.3 Erstellung des Leitfadens

Ebenso wie der Open Source Genehmigungsprozess in Anwendung auf den Produkt Management Prozess wurde ein Leitfaden für Softwareentwicklerinnen und Softwareentwickler erstellt. Dieser wurde wie folgt definiert:

8.3.1 Einleitung

Im ersten Abschnitt des Leitfadens werden das Ziel, die Zielgruppe, sowie die Relevanz des Leitfadens beschrieben.

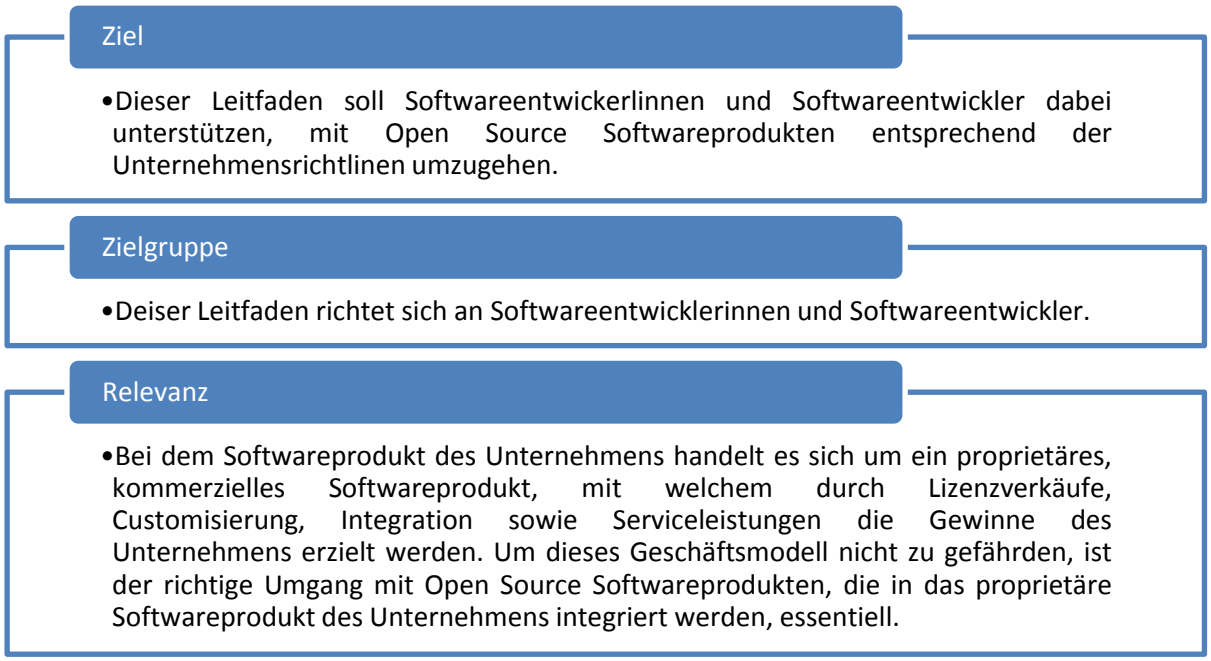

Abbildung 23 - Einleitung des Leitfades

8.3.2 Begriffsdefinitionen

Im folgenden Abschnitt sind Begriffsdefinitionen angeführt, die für das Verständnis der Thematik notwendig sind.

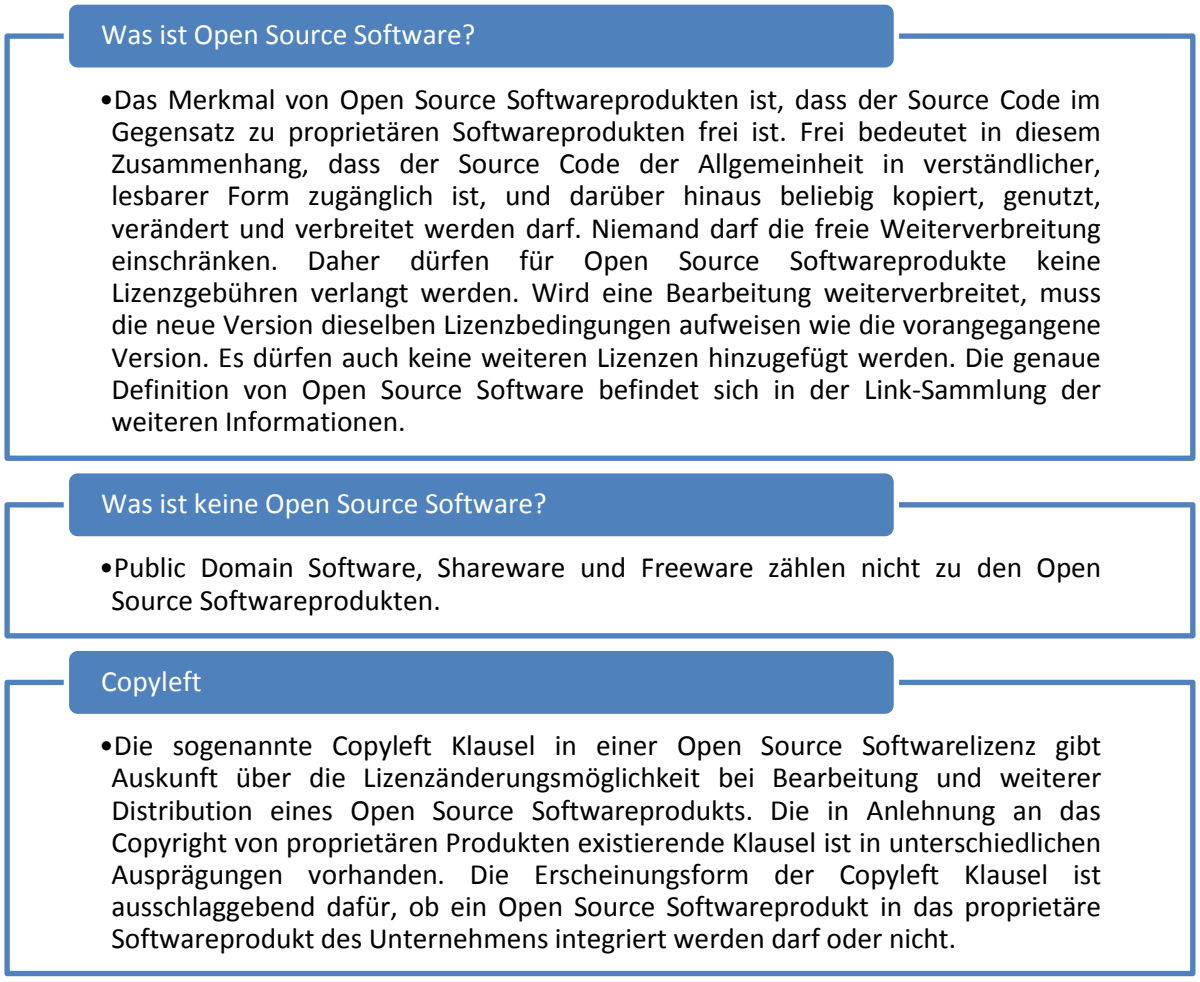

Abbildung 24 - Begriffsdefinition des Leitfades

Die genannten Begriffsdefinitionen können je nach Bedarf erweitert werden.

8.3.3 Lizenzkategorien

Der Abschnitt der Lizenzkategorien gibt Auskunft über die unterschiedlichen Erscheinungsformen der Copyleft Klausel. Bei der Evaluierung eines Open Source Softwareprodukts ist immer zu prüfen, in welcher Form die Copyleft Klausel auftritt. Dies gilt ebenso für Source Code Snippets.

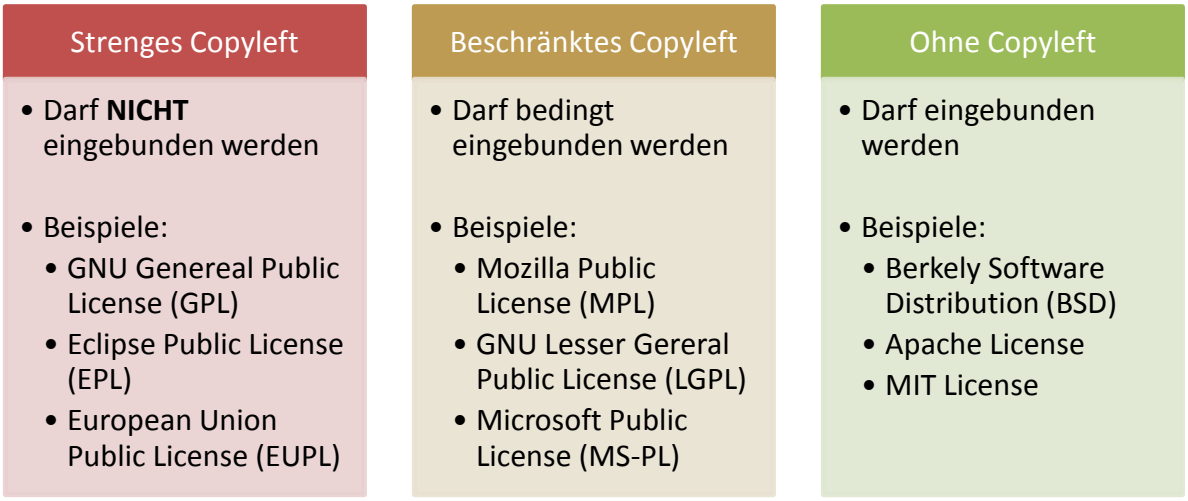

Abbildung 25 - Lizenzkategorien des Leitfadens

8.3.4 Handlungsbedarf

Folglich ist der Handlungsbedarf beschrieben, der bei der jeweiligen Erscheinungsform der Copyleft Klausel gegeben ist.

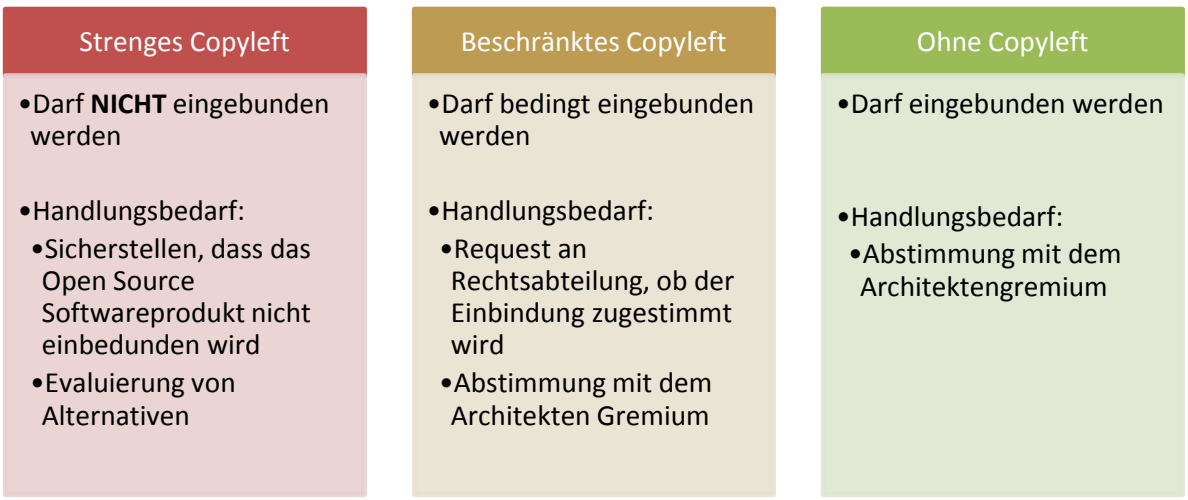

Abbildung 26 - Handlungsbedarf des Leitfadens

Auch bei Open Source Softwareprodukten ohne Copyleft bedarf es der Zustimmung des Architektengremiums, ob das betreffende Produkt tatsächlich in das proprietäre Softwareprodukt des Unternehmens eingebunden werden soll. Dies ist notwendig, da jedenfalls Alternativen und existierende Lösungen zu prüfen sind.

8.3.5 Kontaktinformation

Der Abschnitt der Kontaktinformation gibt Auskunft darüber, an wen man sich bei Fragen zu Open Source Software richten kann. Dies können Wissensträger der jeweiligen Abteilung, sowie die jeweiligen Mitglieder des Architektengremiums sein. Ebenso sind Kontaktinformationen zur jeweiligen Ansprechperson der Rechtsabteilung anzugeben.

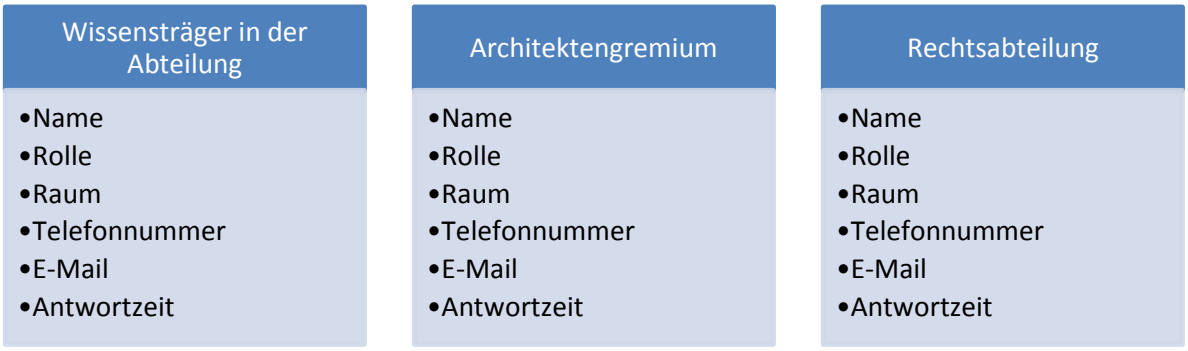

Abbildung 27 - Kontaktinformation des Leitfadens

Die Kontaktinformationen können je nach Abteilung des Unternehmens variieren, und sind daher entsprechend anzupassen.

8.3.6 Weitere Informationen

In diesem Abschnitt befinden sich hilfreiche Links zum Thema Open Source Software.

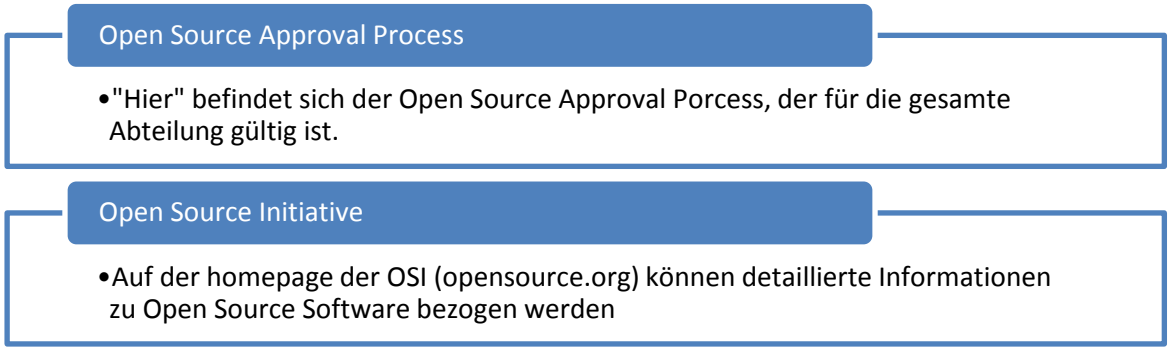

Abbildung 28 - Weitere Informationen des Leitfadens

Dieser Abschnitt kann je nach Bedarf erweitert und angepasst werden.

8.4 Übermittlung der erarbeiteten Artefakte

Im Anschluss an die Festlegung des Open Source Genehmigungsprozesses sowie des Leitfadens wurden diese Artefakte an die Rechtsabteilung des Mutterkonzerns weitergeleitet. Mit dieser wurde vereinbart, den Einsatz der definierten Artefakte mit Unterstützung durch die Rechtsabteilung innerhalb des Unternehmens zu planen und durchzuführen. Des Weiteren sollen die Artefakte und deren Inhalte auf Relevanz für den gesamten Konzern geprüft werden, und

gegebenenfalls in angepasster Form in das Methoden- und Tool-Set des Konzerns aufgenommen werden.

8.5 Fazit

Beim Einsatz von Open Source Softwareprodukten in proprietären Softwareprodukten sind die Erscheinungsform der Copyleft Klausel, sowie die jeweiligen Lizenzbedingungen der Open Source Lizenz genau zu prüfen und entsprechend zu beachten, um das Geschäftsmodell des Unternehmens nicht zu gefährden. Ebenso ist das betreffende Open Source Softwareprodukt von Softwarearchitekten zu prüfen, ob es aus technischer Sicht in das proprietäre Softwareprodukt des Unternehmens eingebunden werden soll. Daher ist es für Unternehmen, die bereits Open Source Softwareprodukte in Verbindung mit ihrem proprietären Softwareprodukt einsetzen bzw. in Zukunft einsetzen wollen, notwendig, einen Prozess zu schaffen, mit dem es möglich ist, den Einsatz von Open Source Softwareprodukten kontrolliert zu steuern. Ebenso müssen Mitarbeiterinnen und Mitarbeiter über den richtigen Umgang mit Open Source Softwareprodukten in Bezug auf das Geschäftsmodell des Unternehmens, für das sie tätig sind, Bescheid wissen, und die Richtlinien des Unternehmens mittragen.

Um dies innerhalb der Produktentwicklungsabteilung eines Softwareunternehmens zu realisieren, wurden Expertinnen und Experten über den IST-Zustand, sowie den SOLL-Zustand im Umgang mit Open Source Softwareprodukten in Bezug auf das proprietäre Softwareprodukt des Unternehmens befragt. Dadurch konnte ein Prozess definiert und dokumentiert werden, der den Umgang mit Open Source Softwareprodukten in den Phasen des Produkt Management Prozesses der Produktenwicklungsabteilung des Unternehmens aufzeigt, sowie alle Rollen sichtbar gemacht werden, die in der jeweiligen Phase betroffen sein können. Ebenso konnte ein Leitfaden für Softwareentwicklerinnen und Softwareentwickler erstellt werden, der als Hilfestellung für die tägliche Entwicklungstätigkeit dient. Die erarbeiteten Artefakte stellen ebenso allgemeine Relevanz dar, da jedes Unternehmen, welches Open Source Softwareprodukte in ihr eigenes, proprietäres Softwareprodukt einbindet, auf den korrekten Umgang mit dieser Thematik achten muss, um sein eigenes Geschäftsmodell nicht zu gefährden. Ebenso können Unternehmen, die den Scrum Framework anwenden, den definierten Prozess als Vorlage für dessen Anwendung und Anpassung im eigenen Unternehmen heranziehen, da Scrum ebenso innerhalb der Produktentwicklungsabteilung des Unternehmens angewandt wird.

9 FOLGESCHRITTE

9.1 Erhebung der Folgeschritte

Diese Folgeschritte wurden im Anschluss an die Fertigstellung und Freigabe der erstellten Artefakte erkannt:

- Ausrollen des Open Source Genehmigungsprozesses innerhalb der Produktentwicklungsabteilung
- Analyse der Produktdokumentation
- Anpassung von weiteren Supportprozessen
- Unternehmensweites Ausrollen

Da laut Aussagen der Expertinnen und Experten die technische Analyse des Source Codes des proprietären Softwareprodukts des Unternehmens bereits durchgeführt wird, stellt dies keinen Folgeschritt dar. Dies gilt ebenso für die Überprüfung der Art und Weise der Dokumentation von Abhängigkeiten. Lediglich bei unternehmensweiten Ausrollen des Prozesses ist zu evaluieren, wie diese Themen in anderen Abteilungen behandelt werden.

9.2 Ausrollen des Open Source Genehmigungsprozesses

Folgende Schritte sind geplant, um Verständnis und Akzeptanz für den neu definierten Prozess zu erlangen:

9.2.1 Bekanntmachung

Um den Open Source Genehmigungsprozess sowie den erstellten Leitfaden innerhalb der Abteilung der Produktentwicklung auszurollen und bekannt zu machen, wird ein Top-Down Ansatz gewählt. Dabei wird als ersten Schritt ein Kick-Off des Open Source Genehmigungsprozess zusammen mit allen Abteilungsleitern und der Geschäftsführung veranstaltet, um für Bewusstseinsbildung und Informationsaustausch innerhalb der Führungsebene zu sorgen. Als zweiten Schritt werden Produktmanager, Softwarearchitekten, und weitere Führungskräfte der Abteilung informiert und geschult, um künftig entsprechend auf den Umgang mit Open Source Softwareprodukten reagieren zu können. Als dritten Schritt werden die bereits informierten und geschulten Führungskräfte zusammen mit den Softwareentwicklerinnen, Softwareentwicklern und Qualitätsingenieuren versammelt, informiert und geschult, um für ein einheitliches Verständnis innerhalb der Abteilung zu sorgen.

9.2.2 Feedback einholen

Nach Vorstellung und Schulung des neuen Prozesses wird nach der Durchführung eines vollständigen Release-Zyklus des proprietären Softwareprodukts des Unternehmens mittels Befragung Feedback von allen Mitarbeiterinnen und Mitarbeitern eingeholt. Dies wird im Rahmen der Release-Retrospektive durchgeführt, in der alle Mitarbeiterinnen und Mitarbeiter der Abteilung teilnehmen.

9.2.3 Überarbeitung

Je nach Bedarf und Feedback der Mitarbeiterinnen und Mitarbeiter der Abteilung werden anschließend Teile des Prozesses sowie des Leitfadens angepasst.

9.3 Anpassung von weiteren Supportprozessen

Die Befragung der Expertinnen und Experten ergab, dass weitere Supportprozesse in Bezug auf den Open Source Genehmigungsprozesses und den allgemeinen Umgang mit Open Source Softwareprodukten anzupassen sind.

9.3.1 Ticketintegration

Um Anfragen zum Thema Open Source in gewohnter Weise abhandeln zu können, soll das bereits bestehende Ticketingsystem um die Kategorie „Open Source Request“ erweitert werden. Dies führt zu höherer Akzeptanz und Durchführung des Prozesses, da der Umgang mit diesem System bereits bekannt ist. Des Weiteren können über diese Abhandlung des Prozesses Ticketvorlagen erstellt werden, die den Anfragesteller bei der Übermittlung der nötigen Unterlagen und Informationen unterstützen.

9.3.2 Trainings

In der bereits bestehenden Basisschulung soll auf den definierten Open Source Genehmigungsprozess hingewiesen werden. Ferner ist die Thematik des gewünschten Umgangs mit Open Source Softwareprodukten in die jährlich verpflichtende Security Schulung und Prüfung zu integrieren. Ebenso kann die Erstellung eines eigenen Onlinetrainings überdacht werden.

9.4 Unternehmensweites Ausrollen

Nach dem Ausrollen des Open Source Genehmigungsprozesses in der Produktentwicklungsabteilung und der potentiell nötigen Prozessüberarbeitung, ist der Open Source Genehmigungsprozess firmenweit auszurollen. Dementsprechend sind die Prozesse der

anderen Abteilungen in Bezug auf den gewünschten Umgang mit Open Source Softwareprodukten ebenso anzupassen.

9.5 Fazit

Mit Festlegung des Open Source Genehmigungsprozesses in Anwendung auf die Abteilung der Produktentwicklung, konnten Folgeschritte wie die Art und Weise des Ausrollens des Prozesses in der Produktentwicklungsabteilung, Anpassung von weiteren Supportprozessen, sowie das unternehmensweite Ausrollen erhoben und näher beschrieben werden. Des Weiteren stellte sich durch die Befragung der Expertinnen und Experten heraus, dass die technische Analyse des proprietären Softwareprodukts des Unternehmens bereits durchgeführt wird, und die Art und Weise der Dokumentation von Abhängigkeiten bereits überprüft ist. Lediglich der definierte Prozess fehlt, der diese Schritte visualisiert und beschreibt. Dementsprechend wurden diese Tätigkeiten innerhalb des Prozesses sichtbar gemacht, und stellen keinen eigenen Folgeschritt dar.

10 ZUSAMMENFASSUNG

Der Begriff Software steht für Computerprogramme und dessen zugehörige Artefakte. Mittels Software kann Hardware gesteuert werden, sie bestimmt also das Verhalten physischer Komponenten. Da Software durch kreative, geistige Schöpfung erzeugt wird, ist der daraus resultierende Source Code, der durch Kompilierung zu einem Softwareprogramm wird, urheberrechtlich geschützt. Software kann auf unterschiedliche Weise gegliedert werden. Für die Beantwortung der Forschungsfrage wurde die Unterteilung nach deren rechtlichen Rahmenbedingungen herangezogen.

Gewerbliche Schutzrechte ermöglichen es, an der eigenen geistigen Schöpfung Eigentum zu begründen. Dies dient dazu, andere von der Nachahmung abzuhalten, und ermöglicht ebenso den Handel mit dem geistigen Eigentum. Gewerbliche Schutzrechte gliedern sich in eingetragene, und nicht eingetragene Schutzrechte. Eingetragene Schutzrechte werden kostenpflichtig in eine allgemein zugängliche Liste aufgenommen, wodurch eine Prüfung dieser Liste möglich ist. Ein nicht eingetragenes Schutzrecht stellt unter anderem das Urheberrecht dar. Dieses existiert automatisch und muss nicht erst in eine Liste aufgenommen werden. Kritischer Faktor ist hierbei die Nachvollziehbarkeit, da es dadurch keine allgemein zugängliche Liste gibt, die überprüft werden kann. Besonders bei Softwareprogrammen ist Augenmerk auf eigenständige Dokumentation zu legen, da man im Streitfall das geistige Eigentum nachweisen können muss. Die Schöpferin bzw. der Schöpfer einer geistigen Leistung hat jedenfalls dafür Sorge zu tragen, gewerbliche Schutzrechte anderer nicht zu verletzen.

Urheberschaft bedeutet Entscheidungsfreiheit über die Art und Weise der Verwertung der eigens erbrachten Leistung. Verwertungsrechte können auch weitergegeben werden, allerdings nicht die Urheberschaft. Die Schutzrechte für Software sind im Urheberrechtsgesetz unter den „Sondervorschriften für Computerprogramme“ definiert. Urheberschaft stellt ein unverzichtbares Recht einer natürlichen Person dar, daher kann diese weder von einer juristischen Person beansprucht, noch an eine andere natürliche oder juristische Person übertragen werden. Die Begünstigung im Sinne des urheberrechtlichen Schutzes von Computerprogrammen wirkt sich einerseits auf alle Staatsangehörige von EU-Staaten aus, andererseits ergibt sich diese durch das Abschließen von Staatsverträgen. Das Recht auf Urheberbezeichnung sowie das Recht auf Werkschutz obliegt dem Dienstgeber, wenn das Werk im Rahmen der dienstlichen Obliegenheit erstellt wird. Dieser kann das Werk nach seinen Vorstellungen benennen und über das öffentliche Erscheinungsbild entscheiden. Für die Verwertung von Computerprogrammen gelten die allgemeinen Verwertungsrechte für Werke laut Urheberrechtsgesetz. Das Vervielfältigungsrecht und das Verbreitungsrecht stellen bei Softwareprogrammen die wirtschaftlich bedeutsamsten Rechte dar. Das Copyright stellt das amerikanische Gegenstück zum europäischen Urheberrecht dar, wobei bei diesem jegliche Rechte übertragbar sind.

Das Wesen proprietärer Softwareprodukte liegt in der Entscheidungsfreiheit der Urheberin bzw. des Urhebers, auf welche Art und Weise und in welchem Umfang das Softwareprodukt verwertet wird. Der Source Code proprietärer Software ist demzufolge meist nur in binärer Form zugänglich und Nutzerinnen und Nutzern die Veränderung und Weiterverbreitung des Softwareprodukts

untersagt. Ebenso verfolgen Unternehmen mit proprietären Softwareprodukten im Regelfall kommerzielle Ziele, daher werden diese Begrifflichkeiten in der Praxis oft in Verbindung gebracht oder als Synonym verwendet. Über eine Lizenz kann definiert werden, welche Nutzungsrechte an Dritte weitergegeben werden, welche geschützt werden, und welche Strafen bei Verletzung der Lizenzbedingungen zu erwarten sind.

Das Wesen eines Open Source Softwareprodukts ist der freie Source Code. Mit frei ist dabei gemeint, dass dieser allen uneingeschränkt zugänglich ist, sowie weiterbearbeitet und weiterverbreitet werden darf. Die Free Software Foundation sowie die Open Source Initiative bilden die Institutionen, welche die Bedingungen freier Software genauer definieren, Open Source Softwarelizenzen herausgeben und sich für die Einhaltung der inkludierten Bedingungen einsetzen. Der frei zugängliche Source Code, die nicht-kommerzielle Einstellung, die gemeinschaftliche Entwicklungsleistung der Community an Softwareentwicklerinnen und Softwareentwicklern, die sich am betreffenden Open Source Softwareprodukt beteiligen, sowie deren dezentralisiertes Arbeiten, fassen die Besonderheiten von Open Source Softwareprodukten zusammen.

Die verschiedenen Open Source Softwarelizenzen gliedern sich primär in Lizenzen mit strengem Copyleft, beschränktem Copyleft, und ohne Copyleft. Der Begriff des Copylefts beschreibt dabei die Lizenzänderungsmöglichkeit bei Bearbeitung und weiterer Distribution eines Open Source Softwareprodukts. Ein strenges Copyleft hat zur Folge, dass jegliche Weiterverbreitung einer Bearbeitung eines Open Source Softwareprodukts unter denselben Lizenzbedingungen zu erfolgen hat, wie das ursprüngliche Open Source Softwareprodukt. Bei Lizenzen mit beschränktem Copyleft ist es unter Einhaltung von Auflagen möglich, das betreffende Open Source Softwareprodukt mit Softwareprodukten, welche unter anderen Lizenzbedingungen stehen, zu kombinieren. Lizenzen ohne Copyleft Klausel erlauben jegliche Einbindung in unfreie Eigenentwicklungen. Ferner können Bearbeitungen des betreffenden Open Source Softwareprodukts proprietär weiterverbreiten werden. In jedem Fall ist beim Einsatz von mehreren Open Source Softwareprodukten mit unterschiedlichen Lizenzen deren Kompatibilität untereinander zu prüfen.

Der Einsatz von Open Source Softwareprodukten bietet Vorteile wie Nachteile. Bei den Vorteilen handelt es ich um Anpassbarkeit, Erweiterbarkeit, Einsparung von Ressourcen, Unabhängigkeit, Offene Standards, Sicherheit, Qualität, Stabilität, sowie Support durch die Community. Dem gegenüber stehen die Nachteile, welche sich in den Bereichen der Wartungskosten, des Produktsupports, der Entwicklungszentrierung, des Schulungsaufwands, der Interoperabilität mit anderen Produkten und der rechtlichen Risiken wiederfinden, wobei letztere hinsichtlich der Forschungsfrage den relevantesten Faktor darstellen.

Unternehmen können im Zusammenhang mit Open Source Softwareprodukten unterschiedliche Positionierungsstrategien verfolgen, indem sie Open Source Softwareprodukte als Hauptgeschäftszweig erzeugen, Komplemente zu Open Source Softwareprodukten herstellen, Open Source Software für interne Prozesse verwenden, oder Open Source Softwareprodukte in eigene proprietäre Softwareprodukte einbinden. Je nach Positionierung können zu den bereits bekannten Vor- und Nachteilen weitere mögliche Vor- und Nachteile für Unternehmen auftreten.

Mögliche Vorteile können Steigerung des Marktanteils, Möglichkeit der Querfinanzierung durch Komplemente, positive Reputation, Behebung von Sicherheitslücken sowie Ressourcenersparnisse darstellen. Denen gegenüber stehen mögliche Nachteile wie Lizenzbindung, Investitionsunsicherheit, Wettbewerbsverluste, sowie Einschränkungen hinsichtlich der möglichen Integrierbarkeit in proprietäre Softwareprodukte und damit verbundene potentielle Gefährdungsmöglichkeiten hinsichtlich des Geschäftsmodells des Unternehmens.

In Bezug auf die Beantwortung der Forschungsfrage ist jene Positionierungsmöglichkeit relevant, bei der Unternehmen Open Source Softwareprodukte in ihre eigenen, proprietären Softwareprodukte integrieren. Aufgrund der Aufbereitung des theoretischen Hintergrunds ergibt sich, dass Open Source Softwareprodukte mit strengem Copyleft nicht in proprietäre Softwareprodukte eingebunden werden dürfen, wenn diese proprietär bleiben sollen. Auch die Einbindung von Open Source Softwareprodukten mit beschränktem Copyleft bringt Auflagen mit sich, die eingehalten werden müssen, um die Lizenzbedingungen nicht zu verletzen. Dementsprechend beschäftigen sich Gerichte und Interessensvertreter der Open Source Community mit Fällen, die die Einbindung von Open Source Softwareprodukten unter den genannten Open Source Softwarelizenzen in proprietäre Softwareprodukte betreffen. Die Mehrheit an Streitfällen hinsichtlich der rechtmäßigen bzw. unrechtmäßigen Verwendung von Open Source Softwareprodukten wird außergerichtlich geregelt. Aus der Analyse der behandelten Präzedenzfälle ergibt sich, dass die klagende Partei zumeist aus dem Umfeld der Open Source Softwareentwicklung stammt. Genauer können dies Softwareentwicklerinnen und Softwareentwickler sein, die selbst an der Entwicklung eines Open Source Softwareprojektes mitgewirkt haben und die Einhaltung der Lizenzbedingungen des entsprechenden Softwareprodukts durchsetzen wollen, oder Vereinigungen und Organisationen, die hinter dem Modell der Open Source Softwareentwicklung stehen, bilden die klagende Partei. Eine weitere Klägergruppe stellen Unternehmen dar, die Wettbewerbsnachteile erfahren, die durch Komplemente anderer Hersteller verursacht werden. Jedenfalls ist bei einer Verurteilung mit Unterlassungsansprüchen und Schadenersatzleistungen zu rechnen, was eine Gefährdung des Geschäftsmodells des Unternehmens darstellt.

Die Antwort auf die Forschungsfrage, welche Aspekte bei der Verwendung von Open Source Softwareprodukten in kommerziellen Softwareprodukten zu berücksichtigen sind, um das eigene Geschäftsmodell nicht zu gefährden, kann somit in dem Sinne beantwortet werden, dass es sich für Unternehmen, die Open Source Softwareprodukte in ihre proprietären Softwareprodukte einbinden, empfiehlt, einen Prozess zu definieren, mit dem es möglich ist, die Einbindung von Open Source Softwareprodukten kontrolliert zu steuern. Ebenso müssen Mitarbeiterinnen und Mitarbeiter über den richtigen Umgang mit Open Source Softwareprodukten in Bezug auf das Geschäftsmodell des Unternehmens, für das sie tätig sind, Bescheid wissen, um die Richtlinien des Unternehmens mittragen zu können. Dessen Realisierung widmet sich der praktische Teil dieser Arbeit, der im Rahmen der Produktentwicklungsabteilung eines Softwareentwicklungsunternehmens durchgeführt wurde.

Als Methodik wurden Expertinnen und Experten des Unternehmens über den IST-Zustand, sowie den SOLL-Zustand in Bezug auf die Einbindung von Open Source Softwareprodukten in das eigene, proprietäre Softwareprodukt des Unternehmens befragt. Dadurch konnte ein Open

Source Genehmigungsprozess definiert und dokumentiert werden, der den Umgang mit Open Source Softwareprodukten in den Phasen des Produkt Management Prozesses der Produktentwicklungsabteilung aufzeigt, sowie alle Rollen sichtbar gemacht werden, die in der jeweiligen Phase betroffen sein können. Ebenso konnte ein Leitfaden für Softwareentwicklerinnen und Softwareentwickler erstellt werden, der als Hilfestellung für die tägliche Entwicklungstätigkeit dient. Die erarbeiteten Artefakte sind ebenso von allgemeiner Relevanz, da jedes Unternehmen, welches Open Source Softwareprodukte in ihr eigenes, proprietäres Softwareprodukt einbindet, auf den korrekten Umgang mit dieser Thematik achten muss, um sein eigenes Geschäftsmodell nicht zu gefährden. Ferner können Unternehmen, die den Scrum Framework anwenden, den definierten Prozess als Vorlage für dessen Anwendung und Anpassung im eigenen Unternehmen heranziehen, da Scrum ebenso innerhalb der Produktentwicklungsabteilung des Unternehmens angewandt wird.

Mit Festlegung des Open Source Genehmigungsprozesses in Anwendung auf die Produktentwicklungsabteilung des Unternehmens konnten Folgeschritte wie die Art und Weise des Ausrollens des Prozesses in der Produktentwicklungsabteilung, Anpassung von weiteren Supportprozessen, sowie das unternehmensweite Ausrollen erhoben und näher beschrieben werden.

Bei der Erstellung des Exposés wurde bei der Zielsetzung, sowie bei der Definition der Methode und des Ablaufs davon ausgegangen, dass ein initialer Open Source Genehmigungsprozess durch die Rechtsabteilung des Mutterkonzerns bereits auf alle Standorte ausgerollt wurde, und dessen Anwendung in Bezug auf Bekanntheitsgrad und Verbesserungspotentiale analysiert werden kann. Da die Erstellung eines solchen initialen unternehmensweiten Prozesses auf bislang unbekannten Zeitpunkt verschoben wurde, mussten ebenso die Zielsetzung, sowie die Definition der Methode und des Ablaufs auf diese Gegebenheit angepasst werden.

ANHANG A - The Open Source Definition

Die Open Source Initiative listet laut (OSI, 2007) folgende Kriterien für Open Source Softwareprodukte:

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. *Free Redistribution*

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. *Source Code*

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. *Derived Works*

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. *Integrity of The Author's Source Code*

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. *No Discrimination Against Persons or Groups*

The license must not discriminate against any person or group of persons.

6. *No Discrimination Against Fields of Endeavor*

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. *Distribution of License*

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. *License Must Not Be Specific to a Product*

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. *License Must Not Restrict Other Software*

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. *License Must Be Technology-Neutral*

No provision of the license may be predicated on any individual technology or style of interface.

ABKÜRZUNGSVERZEICHNIS

ADR – Applied Data Research
AGB – Allgemeine Geschäftsbedingungen
AKV – Aufgaben, Kompetenzen, Verantwortlichkeiten
AL – (Perl) Artistic License
BSD – Berkeley Software Distribution
EULA – End User License Agreement
EUPL – European Union Public License
FSF – Free Software Foundation
FSFE – Free Software Foundation Europe
GNU – Rekursives Akronym für „GNU is not Unix“
GMG – Gebrauchsmustergesetz
GPL – General Public License
IBM – International Business Machine Corporation
MIT – Massachusetts Institute of Technology
MPL – Mozilla Public License
MarkenSchG - Markenschutzgesetz
MS-PL – Microsoft Public License
MuSchG – Musterschutzgesetz
OS – Open Source
OSI – Open Source Initiative
OSL – Open Source Lizenz
OSS – Open Source Software
PatG – Patentgesetz
SFLC – Software Freedom Law Center
UrhG – Urheberrechtsgesetz

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LITERATURVERZEICHNIS

Adams, S. (25. 02 2013). Dilbert ©2013, Universal Uclick.

AustroMechana. (2016). *www.akm.at*. Von http://www.akm.at/service/urheberrecht/ abgerufen

Bdeiwi, S. (2016). Von eStrategy-Magazins: http://www.estrategy-magazin.de/open-source-software-rechtliche-hintergruende-sowie-chancen-und-risiken-fuer-unternehmen.html abgerufen

BlackDuck. (2016). *www.blackducksoftware.com*. Von https://www.blackducksoftware.com/top-open-source-licenses abgerufen

Bridge, R. (04. 11 2013). *entrepreneurhandbook.co.uk*. Von http://entrepreneurhandbook.co.uk/open-source-software/ abgerufen

Brückner, R. M. (2013). *Erfolgreiche Software-Lizenzierung: Electronic License Management - Von Der Auswahl Bis Zur Installation .* Springer.

Bruegge, B., Harhoff, D., Picot, A., Creighton, O., Fiedler, M., & Henkel, J. (2004). *Open-Source-Software - eine ökonomische und technische Analyse.* Springer.

Brügge, B. (2012). *Open-Source-Software: Eine ökonomische und technische Analyse (German Edition).* Springer.

Capers, J. (2014). *The Technical and Social History of Software Engineering.* Addison-Wesley.

Cruz, S. (o6. 07 2012). *http://www.i95dev.com/*. Von http://www.i95dev.com/open-source-communities-an-optimal-solution-for-smbs/ abgerufen

Deister, J., & Meyer-Spasche, G. (2009). *Anwaltsstrategien im Software-Recht.* Heilborn, Köln: Richard Boorberg Verlag.

Diedrich, O. (24. 07 2007). *heise.de*. Von https://www.heise.de/newsticker/meldung/Urteil-gegen-Skype-wegen-GPL-Verletzung-154544.html abgerufen

Diedrich, O. (09. 08 2016). *heise.de*. Von https://www.heise.de/newsticker/meldung/GPL-Klage-gegen-VMware-abgewiesen-3291188.html abgerufen

Dölle, M. (26. 06 2013). *heise.de*. Von https://www.heise.de/newsticker/meldung/GPL-Urteil-Quellen-und-Binaries-muessen-gleiche-Version-haben-1897161.html abgerufen

Ecomsolutions. (12 2007). *blog.ecomsolutions.ne*. Von http://blog.ecomsolutions.net/2007/12/disadvantages-of-open-source-software/ abgerufen

Faber, M. J. (2008). *Open Innovation: Ansätze, Strategien und Geschäftsmodelle (Spektrum wirtschaftswissenschaftliche Forschung).* Gabler Verlag.

Fina, S. (2006). *www.univie.ac.at.* Von http://spl.univie.ac.at/fileadmin/user_upload/inst_unternehmensrecht/Wahlfachkorb_Technologierecht/SS_06/Fina_Software-Schutz.pdf abgerufen

Fogel, K. (06. 03 2013). *http://producingoss.com/*. Von http://producingoss.com/de/license-compatibility.html abgerufen

FreeSoftwareFoundation. (14. 9 2016). *gnu.org*. Von http://www.gnu.org/philosophy/categories.de.html#non-freeSoftware abgerufen

Freund, T. (2006). *Software Engineering durch Modellierung wissensintensiver Entwicklungsprozesse.* GITO.

FSFE. (26. 06 2013). *fsfe.org*. Von https://fsfe.org/news/2013/news-20130626-01.de.html abgerufen

Groll, T. (2015). *1x1 des Lizenzmanagements: Praxisleitfaden für Lizenzmanager, 3. Auflage.* arl Hanser Verlag GmbH & Co. KG.

Haruvy, E., Sethi, S. P., & Zhou, J. (02. 01 2008). Open Source Development with a Commercial Complementary Product or Service. Dallas.

Hennig, S. (2009). *Open source-Software für mittelständische Unternehmen: Eine betriebswirtschaftliche Analyse, 1. Auflage.* Igel Verlag Fachbuch.

Herzwurm, G. (2006). Computer - Software. Stuttgart.

Ihlenfeld, J. (14. 04 2005). *www.golem.de*. Von https://www.golem.de/0504/37499.html abgerufen

Ihlenfeld, J. (24. 07 2007). *golem.de*. Von https://www.golem.de/0707/53684.html abgerufen

Ihlenfeld, J. (20. 06 2011). *golem.de*. Von https://www.golem.de/1106/84313.html abgerufen

Jaeger, T., & Metzger, A. (2016). *Open Source Software: Rechtliche Rahmenbedingungen der Freien Software; 4. Auflage.* C.H.Beck.

Jäger, T. (15. 12 2008). *ifross.org*. Von http://www.ifross.org/artikel/fsf-gegen-cisco-gericht-wegen-gpl-und-lgpl-verletzungen abgerufen

Koglin, O. (09. 10 2006). *www.ifross.org*. Von http://www.ifross.org/artikel/urteil-des-landgericht-frankfurt-am-gpl abgerufen

Krcmar, H. (2005). *Informationsmanagement 4. Auflage.* Springer.

Kueng, J. (25. 05 2009). *ifross.org*. Von http://www.ifross.org/artikel/einigung-im-gpl-streit-zwischen-fsf-und-cisco-systems abgerufen

Küng, J. (11. 05 2008). *fross.org/*. Von http://www.ifross.org/artikel/urteil-gegen-skype-rechtskraeftig abgerufen

Labesius, S. (20. 10 2016). *ifross.org*. Von http://www.ifross.org/artikel/hellwig-vmware-landgericht-hamburg-h-lt-urheberrechte-f-r-nicht-belegt abgerufen

Landley, R. (23. 05 2009). *landley.net*. Von http://landley.net/notes-2009.html abgerufen

Lassmann, W. (2006). *Wirtschaftsinformatik. Nachschlagewerk für Studium und Praxis .* Gabler.

Lichter, H., & Ludewig, J. (2013). *Software Engineering: Grundlagen, Menschen, Prozesse, Techniken.* dpunkt-Verlag.

Lindner, M. (04. 08 2010). *pro-linux.de*. Von https://www.pro-linux.de/news/1/16001/busybox-gewinnt-weiteren-streit-um-die-gpl.html abgerufen

Lipinski, K. (2016). *ITWissen.info*. Von http://www.itwissen.info/definition/lexikon/Software-SW-software.html abgerufen

Moore, J. (Regisseur). (2001). *Revolution OS* [Kinofilm].

Müller, A. (04. 08 2010). *heise.de*. Von https://www.heise.de/newsticker/meldung/Busybox-Entwickler-siegen-vor-Gericht-gegen-Westinghouse-Digital-Technologies-1050544.html abgerufen

OSI. (22. 03 2007). *Opensource.org*. Von https://opensource.org/osd abgerufen

Roger, B. (13. 08 2010). *ifross.org*. Von http://www.ifross.org/artikel/erste-streitige-durchsetzung-gpl-usa abgerufen

Roger, B. (29. 11 2011). *ifross.org*. Von http://www.ifross.org/artikel/lg-berlin-veraenderung-gpl-software-routers-grundsaetzlich-zulaessig-fritzbox abgerufen

Rouse, M. (03 2011). *whatis.techtarget.com*. Von http://whatis.techtarget.com/definition/MIT-License-X11-license-or-MIT-X-license abgerufen

Rouse, M. (06 2014). *whatis.techtarget.com*. Von http://whatis.techtarget.com/definition/fork abgerufen

Rouse, M. (05 2016). *searchenterprisesoftware.de*. Von http://www.searchenterprisesoftware.de/definition/Apache-Lizenz abgerufen

Ruffing, B. (13. 02 2017). *prozessmaler.de*. Von http://prozessmaler.de/die-3p-die-drei-wesentlichen-eckpfeiler-zur-organisation-von-geschaeftsprozessen/ abgerufen

Schaaf, A. (2013). *Open-Source-Lizenzen: Untersuchung der Gpl, Lgpl, Bsd und Artistic License.* Diplomica Verlag.

Schraml, K. (25. 11 2014). *blog.pascom.net*. Von http://blog.pascom.net/de/voip-fuer-business/telefonanlagen-ratgeber-propreitaeres-vs-opensource-system/ abgerufen

Schrape, J.-F. (02 2015). Open source Softwareprojekte zwischen Passion und Kalkül. Stuttgart.

Smith, J. (10. 05 2016). *oreilly.com/i*. Von https://www.oreilly.com/ideas/using-open-source-methods-for-internal-software-projects abgerufen

Storz, S. (22. 05 2012). *eStrategy-Magazin*. Von http://www.estrategy-magazin.de/ausgabe-04-2011/open-source-vs-proprietaere-content-management-systeme-welche-strategie-wann.html abgerufen

usedSoft. (2016). *usedSoft Deutschland GmbH*. Von https://www.usedsoft.com/de/gebrauchte-software/usedsoft-wiki/glossar/lizenzrecht/softwarelizenzen-recht/ abgerufen

Vaughan-Nichols. (31. 07 2013). *blog.smartbear.com*. Von http://blog.smartbear.com/open-source/using-open-source-methods-in-a-private-company/ abgerufen

Weber, S. (2005). *The Success of Open Source.* Harvard Univ Pr.

Welte, H. (14. 04 2005). *gpl-violations.org*. Von http://gpl-violations.org/news/ abgerufen

Welte, H. (22. 09 2006). *gpl-violations.org*. Von http://gpl-violations.org/news/ abgerufen

Wilkens, A. (14. 04 2005). *www.heise.de*. Von https://www.heise.de/newsticker/meldung/Einstweilige-Verfuegung-gegen-Fortinet-wegen-Verstosses-gegen-die-GPL-153026.html abgerufen

Williams, S. (2002). *Free as in Freedom - Richard Stallman´s Crusade for Free Software.* O`Reilly.

Wilson, S. (05. 02 2015). *osswatch.jiscinvolve.org*. Von https://osswatch.jiscinvolve.org/wp/2015/02/05/open-source-software-licensing-trends/ abgerufen

WKO. (31. 01 2017). *wko.at*. Von https://www.wko.at/Content.Node/Service/Innovation-und-Technologie/Patent-Marke-Muster/Einfuehrung-in-gewerbliche-Schutzrechte.html abgerufen