Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Finanzdepartement EFD

Informatiksteuerungsorgan des Bundes ISB

Praxis-Leitfaden

Open Source Software in der Bundesverwaltung

IKT-Empfehlung zur Bundesinformatik[1]

Klassifizierung:[2]	Nicht klassifiziert
Verbindlichkeit:[3]	Empfehlung
Planungsfeld:[4]	IKT der Bundesverwaltung
Diese Version:	1.0 vom 19.12.2019
Ersetzt Version:	Ohne Vorversion
Status:	Genehmigt
Beschlussdatum / Datum der Inkraftsetzung (diese Version):	Dieser Leitfaden ist als Hilfsmittel zu verstehen und benötigt deshalb keinen formellen Beschluss. Dadurch kann das Dokument einfacher laufend ergänzt oder angepasst werden.
Freigegeben durch, Rechtsgrundlage:	Informatiksteuerungsorgan des Bundes (ISB), gestützt auf Artikel 17 Absatz 1 der Verordnung vom 9. Dezember 2011 über die Informatik und Telekommunikation in der Bundesverwaltung (BinfV), SR 172.010.58
Sprachen:	Deutsch

[1] «IKT-Empfehlung» gemäss [P035], *Abschnitt 4.7*

[2] Zu der Klassifizierung INTERN und VERTRAULICH vgl. *2. Abschnitt Verordnung vom 4. Juli 2007 über den Schutz von Informationen des Bundes, SR 510.411*

[3] Vgl. Fussnote 1

[4] Planungsfelder gemäss *IKT-Strategie des Bundes 2016-2019 vom 4. Dezember 2015, Anhang A (SB000)*

Inhaltsverzeichnis

1. Definitionen 3
2. Potenzial und Herausforderungen 4
2.1 Potenzial von Open Source Software 4
2.2 Herausforderungen im Umgang mit Open Source Software 5
3. Konstellationen der Nutzung von Open Source Software 7
3.1 Einsatz unveränderter Open Source Software 8
3.2 Entwickeln mit Open Source Komponenten 9
3.3 Freigabe von Open Source Quellcode 9
4. Eigenschaften von Open Source Lizenzen 10
4.1 Copyleft 10
4.2 Open Source Lizenzen mit starkem Copyleft (Strong Copyleft) 10
4.3 Open Source Lizenzen mit schwachem Copyleft (Weak Copyleft) 11
4.4 Permissive Open Source Lizenzen (No Copyleft) 11
4.5 Kompatibilität von Open Source Lizenzen 11
4.6 Weiterführende Informationen zu rechtlichen Fragestellungen 12
5. Geschäftsmodelle mit Open Source Software 13
5.1 Services und Produkte basierend auf Open Source Software 13
5.2 Services für Open Source Software 14
5.3 Subscriptions 14
5.4 Dual Licensing 14
6. Analysemöglichkeiten von Open Source Software 14
6.1 alternativeTo 15
6.2 GitHub 15
6.3 Open Hub 16
7. Support-Modelle beim Einsatz von Open Source Software 17
7.1 Einsatz ohne professionellen Support 17
7.2 Einsatz mit internem Support 17
7.3 Einsatz mit Support durch externen Anbieter 18
8. Freigabe von Open Source Software 18
9. Fragen aus der Praxis 18
10. Weiterentwicklung des Praxisleitfadens 20
Referenzen 21

1. Definitionen

Open Source Software	Eine Software wird als Open Source Software (OSS) bezeichnet, wenn sie unter einer der rund 80 Lizenzen veröffentlicht ist, welche durch die Open Source Initiative anerkannt sind (Open Source Initiative 2019). Im Detail gibt die «Open Source Definition» zehn Kriterien vor, welche alle Open Source Lizenzen zu erfüllen haben (Perens 1999). Im Wesentlichen geben diese die folgenden vier Bedingungen vor, welche auch die Definition von Freier Software umfassen: 1. Die Software darf beliebig eingesetzt werden. 2. Der Quellcode der Software wird zugänglich gemacht. 3. Die Software darf beliebig kopiert und verbreitet werden. 4. Die Software darf verändert und in veränderter Form unter bestimmten Bedingungen weitergegeben werden.
Freie Software	Noch vor dem Begriff «Open Source» wurde von der Free Software Foundation (FSF) in den 80er-Jahren der Begriff der «Freien Software» eingeführt. Inhaltlich entspricht Freie Software den Bedingungen von Open Source Software, zielt jedoch darauf ab, dass die Software stets frei zugänglich ist und möglichst nicht in proprietäre Software integriert wird.[5]
Proprietäre Software	Bei einer proprietären Software entwickelt ein Anbieter eine Software und verkauft eine Nutzungslizenz an den Anwender. Der Anwender kennt in der Regel den Software-Code nicht und darf die Software nicht weitergeben oder ändern. Der Anwender kann die Software nur gemäss den Lizenzvorgaben (z.B. End-User License Agreement, EULA) gegen Bezahlung von Lizenzgebühren nutzen: z.B. darf die Software nur von einer bestimmten Anzahl Nutzern verwendet werden oder nur für eine bestimmte Anzahl von Prozessoren.[6] Für die meist jährlich anfallenden Wartungsgebühren verpflichtet sich der Anbieter innert nützlicher Frist Fehler zu beheben und die Software kontinuierlich weiter zu entwickeln.

[5] Siehe https://de.wikipedia.org/wiki/Freie_Software

[6] Siehe https://de.wikipedia.org/wiki/Propriet%C3%A4re_Software

2. Potenzial und Herausforderungen

Das Potenzial von Open Source Software in der heutigen Informatik ist hoch. Gleichzeitig bringt die Nutzung von Open Source Lösungen aber auch Herausforderungen mit sich. Diese zwei Seiten werden im nachfolgenden Abschnitt näher erläutert. Die Angaben basieren u.a. auf den Resultaten der Open Source Studie Schweiz 2018 der Universität Bern, bei der die Befragten Auskunft über die erkannten Vor- und Nachteile von Open Source Software erteilten (Stürmer und Gauch 2018).

2.1 Potenzial von Open Source Software

Die folgenden Punkte beschreiben das Potenzial, das bei der Verwendung und der Freigabe von Open Source Software realisiert werden kann.

1. Keine Lizenzgebühren	Für die Nutzung von Open Source Software fallen keine Lizenzkosten an. Beim Bezug von komplexen Standard Open Source Software-Paketen kann jedoch der Bezug von so genannten «Subscriptions» (Support-Abo) Sinn machen, für die eine Gebühr bezahlt wird.
2. Kostenersparnisse durch Kooperationen mit anderen Nutzern	Da Software unter einer Open Source Lizenz unbeschränkt genutzt und weiterentwickelt werden kann, können nach dem Prinzip «Einmal entwickeln – mehrfach anwenden» die Kosten für Weiterentwicklungen geteilt bzw. bestehende Zusatzentwicklungen von anderen Verwaltungsstellen übernommen werden. Gleichzeitig besteht die Möglichkeit von Erfahrungen und Entwicklungen anderer zu profitieren.
3. Community Building & Wissensaustausch	Open Source Software erleichtert das Community Building und den Wissensaustausch beispielsweise zwischen den verschiedenen föderalen Ebenen der Schweiz. Es kann gemeinsam an einer Software weiterentwickelt, Fehler behoben und individuelle Erfahrung geteilt werden. Der gesteigerte Wissensaustausch zwischen verschiedenen Verwaltungsstellen führt dazu, dass ein besseres Verständnis davon entsteht, woran andere arbeiten, so dass Doppelspurigkeiten vermieden werden und sich die besten Lösungen verbreiten.
4. Niedrigere Herstellerabhängigkeit	Die Abhängigkeit von Software-Herstellern (Vendor Lock-in) wird in der Informatik als sehr hoch eingeschätzt. Die Verwendung von Software unter einer Open Source Lizenz reduziert die Abhängigkeit von den Herstellern da Betrieb, Wartung, Support, Weiterentwicklung und andere Dienstleistungen für Open Source Software offen ausgeschrieben werden können.
5. Offene Standards und hohe Interoperabilität	Bei Open Source Software ist die Kompatibilität mit anderen Softwarelösungen und Informatiksystemen (Interoperabilität) in der Regel höher als bei proprietärer Software. Auch verwenden Open Source Lösungen praktisch ausschliesslich offene Datenformate weshalb diese einfach mit anderen System ausgetauscht werden können.

6. Transparenz über den Aufbau der Software	Da die Software auch in Form des Source Codes vorliegt, kann ihre Qualität z.B. durch externe Reviews geprüft werden. Zudem lassen sich anhand des Quellcodes Dokumentationen (z.B. im Hinblick auf Neuausschreibungen für Weiterentwicklungsleistungen oder die Ablösung einer Open Source Software am Ende der Lebensdauer) erstellen.
7. Sicherheit und Vertrauen durch Transparenz	Durch die Offenheit des Quellcodes kann die Sicherheit von Open Source Software höher sein als bei proprietäre Software. Ausserdem ist es bei Open Source Software nahezu unmöglich, Hintertüren und andere Lücken bewusst einzubauen was zu mehr Vertrauen in die Software führt.
8. Höhere Qualität und Modularität des Quellcodes	Die Qualität von Open Source Software kann höher sein als bei proprietärer Software, da die Motivation guten Programmier-Code zu schreiben höher sein kann, wenn die Entwickler wissen, dass ihr Quellcode veröffentlicht werden wird. Zudem sind Open Source Lösungen typischerweise hoch modular aufgebaut, so dass einzelne Module einfach ersetzt und zugleich die übrigen weiterverwendet werden können.
9. Einfache Anpassungen an eigene Bedürfnisse	Der Zugriff auf den Quellcode ermöglicht es Nutzern, selber oder durch externe Anbieter Weiterentwicklungen vorzunehmen. So kann die Software rasch den eigenen Bedürfnissen angepasst werden.
10. Rasche Innovation und Integration	Es findet eine schnelle, kontinuierliche Weiterentwicklung von Open Source Software durch die Community statt. Neue Technologien und Datenstandards werden bspw. oft als Open Source Programmier-Bibliotheken veröffentlicht. Dies ermöglicht die rasche Realisierung von innovativen Software-Lösungen.
11. Arbeitgeberattraktivität	Der Einsatz von modernen Open Source Technologien und die informelle Zusammenarbeit mit internationalen Communities fördert die Motivation der Mitarbeitenden und erhöht dadurch die Arbeitgeberattraktivität, was wiederum die Rekrutierung von jungen, qualifizierten Fachkräften erleichtert.

2.2 Herausforderungen im Umgang mit Open Source Software

Im Folgenden werden die typischen, in der Praxis angetroffenen Herausforderungen mit Open Source Software beschrieben und mögliche Lösungsansätze aufgezeigt.

1. Hohe Wechselkosten aufgrund bestehender Abhängigkeiten	Eine Ablösung von proprietärer Software durch Open Source Lösungen kann sehr aufwändig sein, wenn die Applikation über Schnittstellen-Integration oder anderen Abhängigkeiten eng in das bestehende Informatik-System eingebettet ist. Diese Wechselkosten führen dazu, dass sich Einführungen von Open Source Software oftmals nur beim Ablauf des Lebenszyklus der proprietären Software lohnen.

2. Fehlende Features oder keine passende Open Source Lösung	Soll eine Standardsoftware beschafft werden, kann es vorkommen, dass zur (bisherigen) proprietären Lösung keine oder nur eine unzureichend funktionale Open Source Software Alternative zur Verfügung steht. Als Lösungsansatz besteht die Möglichkeit, zusammen mit anderen Nutzern der Open Source Software die fehlenden Features gemeinsam zu entwickeln und diese dem Open Source Projekt hinzuzufügen.
3. Höhere Aufwände bei der Integration	Kurzfristige Kostenersparnisse, die durch Lizenzeinsparungen beim Einsatz von Open Source Software anfallen, werden oftmals kompensiert durch höhere Aufwände, die bei der Integration von Open Source Software entstehen. Somit ist es wichtig zu erkennen, dass der professionelle Einsatz von Open Source Software nicht «gratis» ist, sondern internen oder externen Aufwand generiert.
4. Verantwortlichkeiten und Support unklar	Kritisiert wird an Open Source Software oftmals das Fehlen von Support und Wartung durch externen Unternehmen. Zwischenzeitlich gibt es allerdings eine Vielzahl an Anbietern, die kommerzielle Dienstleistungen (langfristigen Support, fortlaufende Wartung, Weiterentwicklung, Haftung und Gewährleistung etc.) für Open Source Lösungen anbieten. Die unterschiedlichen Geschäftsmodelle mit Open Source Software sind in Abschnitt 5 «Geschäftsmodelle mit Open Source Software» erläutert, die verschiedenen Support-Modelle in Abschnitt 7 «Support-Modelle beim Einsatz von Open Source Software» dargestellt. Eine aktuelle Übersicht über Open Source Dienstleister und deren Leistungen sind im Open Source Directory (*https://www.ossdirectory.com*) zu finden.
5. Teilweise kleiner Markt mit wenig Anbietern	Da gewisse Open Source Applikationen wie bspw. Desktop-Anwendungen grundsätzlich wenig oder gar keinen Support- und Wartungsaufwand verursachen, besteht in gewissen Fällen kaum ein Markt von Open Source Anbietern für Support- und Wartungsdienstleistungen. In diesem Fall kann eine Ausschreibung von Dienstleistungen für Open Source Software dazu beitragen, dass sich ein Markt bildet, der den Wettbewerb unter den Anbietern stärkt. Im Hinblick auf die Ausschreibung solcher Leistungen sind das Bestehen einer breit abgestützten Entwicklungs-Community für die betreffende Software und die Qualität der Entwicklungsdokumentation von praktischer Bedeutung.
6. Geringe Visibilität	Communities von Open Source Projekten fokussieren sich vorwiegend auf die Produktentwicklung und kaum auf deren Vermarktung. Demgegenüber investieren Hersteller von proprietärer Software intensiv in Marketing und Verkauf ihrer Produkte. Durch diese fehlende Werbung für Open Source Software entsteht oftmals das falsche Bild, dass keine Alternative zu den proprietären Produkten existiert. Es gibt jedoch Plattformen wie alternativeTo, welche einen Überblick über Lösungen für bestimmte Aufgabestellungen bieten (siehe dazu Abschnitt 6).
7. Mangelnde Akzeptanz der Endbenutzer	Aufgrund andersartiger Benutzeroberflächen, fehlender Funktionen, niedriger Benutzerfreundlichkeit und wenig Werbung für Open Source Software besteht bei gewissen Open Source Produkten eine mangelhafte Akzeptanz der Endbenutzer. Entsprechende Kommunikation, Dokumentation und Schulungsangebote können dieser Herausforderung entgegenwirken. Dass sich Open Source Software entsprechend den Bedürfnissen der Benutzer weiterentwickeln lässt, kann ebenfalls zu einer höheren Akzeptanz führen.

8. Wenig oder kein internes Knowhow	Open Source Lösungen entwickeln sich rasch weiter und benötigen gleichzeitig ein vertieftes technisches Verständnis. So kann es vorkommen, dass bei internen Mitarbeitenden für gewisse Open Source Software kein oder unzureichendes internes Knowhow besteht. Durch Weiterbildungen und Möglichkeiten für Selbststudium über Online-Quellen und den Aufbau eigener Wikis etc. kann internes Open Source Knowhow aufgebaut werden.
9. Schwierige Zukunftseinschätzung	Oftmals ist es für Aussenstehende einer Open Source Community schwierig zu erkennen, wie sich das Projekt in Zukunft weiterentwickeln wird. Deshalb ist es wichtig, eine realistische Einschätzung über die künftige Weiterentwicklung einer Open Source Lösung vornehmen zu können. In Abschnitt 6.2 wird diesbezüglich die Plattform Open Hub vorgestellt, die eine Einschätzung bezüglich Aktivität und Heterogenität der Entwickler-Community erlaubt. Dies ermöglicht die zukünftige Entwicklung eines Open Source Projektes besser einzuschätzen.
10. Rechtliche Unsicherheiten	Die Vielzahl unterschiedlicher Open Source Lizenzen und die bisher geringe Anzahl von Gerichtsentscheidungen zu Auslegungsfragen führen mitunter zu rechtlichen Unsicherheiten mit Open Source Software. Einen Überblick über die wichtigsten Open Source Lizenzen und deren Eigenschaften und Kompatibilitäten soll dieser Praxisleitfaden vermitteln. Quellen für vertiefende Antworten zu juristischen Fragenstellungen sind in Abschnitt 4.6 «Weiterführende Informationen zu rechtlichen Fragestellungen» aufgeführt.
11. Zu umfangreiches Angebot an Open Source Software	In den letzten Jahren hat die Anzahl Open Source Software massiv zugenommen. Potenzielle Nutzer von Open Source Software bemängeln deshalb die Vielzahl vorhandener Open Source Lösungen, die es auf dem Markt gibt. Aus diesem Grund werden in Kapitel 6 «Analysemöglichkeiten von Open Source Software» mit alternativeTo und Open Hub zwei Plattformen vorgestellt, die einen Vergleich von Open Source Lösungen ermöglichen.

3. Konstellationen der Nutzung von Open Source Software

Bei der Nutzung von OSS durch die Bundesverwaltung lassen sich im Hinblick auf die Pflichten in Zusammenhang mit Open Source Lizenzen unterschiedliche Konstellationen unterscheiden:

Konstellation	***Auswirkung***
Die **blosse Verwendung** von Open Source Software (ohne Modifikationen des Programmcodes)	Es besteht **keine Verpflichtung** zur Weitergabe des Sourcecodes
Die **vollständige Neuentwicklung** von Software, welche unter einer Open Source Lizenz publiziert werden soll	In diesem Fall besteht **vollständige Freiheit**, unter welcher Lizenz die Software publiziert werden soll.

Das **Weiterentwickeln** (intern oder extern) mit bestehenden Open Source Komponenten (mit oder ohne Copyleft-Effekt), solange die Software **ausschliesslich innerhalb der gleichen Organisation genutzt** und nicht weitergegeben wird	Die OSS Komponenten können beliebig mit Drittkomponenten kombiniert werden. Es besteht grundsätzlich **keine Publikationspflicht solange** die Software **ausschliesslich innerhalb der gleichen Organisation genutzt** und nicht weitergegeben wird. Da rechtlich nicht abschliessend geklärt ist, wie weit der Begriff einer ‚Weitergabe innerhalb der gleichen Organisation' reicht, kommt dieses Szenario im Fall von OSS-Komponenten mit Copyleft-Effekten nur in Ausnahmefällen unter Beizug des jeweiligen Rechtsdienstes in Frage. Siehe dazu auch weiter hinten Ziff. 3.2 und Frage 11 im Abschnitt 8 «Fragen aus der Praxis»
Das **Weiterentwickeln** (intern oder extern) **ausschliesslich mit** bestehenden Open Source **Komponenten ohne Copyleft-Effekt**, wenn die Software auch extern (z.B. an Kantone) weitergegeben werden soll	Die OSS Komponenten können beliebig mit Drittkomponenten kombiniert werden und es besteht **keine Publikationspflicht** (siehe zum Copyleft-Effekt Abschnitt 4 «Eigenschaften von Open Source Lizenzen»).
Das **Weiterentwickeln** (intern oder extern) mit bestehenden Open Source Komponenten **mit Copyleft-Effekt**, wenn die Software auch extern (z.B. an Kantone) weitergegeben werden soll	Es besteht eine **Pflicht zur Publikation unter einer Lizenz mit Copyleft** (siehe zum Copyleft-Effekt Abschnitt 4 «Eigenschaften von Open Source Lizenzen»).

3.1 Einsatz unveränderter Open Source Software

Beim Einsatz von Open Source Software ist grundsätzlich nicht von Relevanz, unter welcher Open Source Lizenz diese veröffentlicht ist, denn sämtliche Open Source Lizenzen (siehe Abschnitt 1 «Definitionen») erlauben die uneingeschränkte Nutzung von Open Source Software, egal auf wie vielen Arbeitsplätzen, gleichzeitig eingeloggten Benutzern, Anzahl Servern und Prozessoren etc. die Software eingesetzt wird. Die unterschiedlichen Eigenschaften der Open Source Lizenzen spielen erst bei der Weitergabe der eigenen Software-Entwicklung über die eigene Organisation hinaus eine Rolle. Diese Software-Entwicklung kann vorbestehenden Open Source Komponenten integrieren (vgl. Abschnitt 3.2 «Entwickeln mit Open Source Komponenten») oder der eigene Quellcode soll als Open Source Software veröffentlicht werden (vgl. Abschnitt 3.3 «Freigabe von Open Source Komponenten» und 4.5 «Kompatibilität von Open Source Lizenzen»).

Beim Einsatz von unveränderter Open Source Software entstehen selbst bei Programmen, welche unter einer Lizenz mit strengem Copyleft-Effekt stehen, keine Publikationsverpflichtungen.

Solange Open Source Software durch interne Mitarbeitende eingeführt und betrieben wird, kann dies ohne öffentliches Beschaffungsverfahren geschehen, denn «Die OSS-Lizenz allein kostet die beschaffende Stelle in der Regel nichts und ist für sich allein daher auch nicht beschaffungsrelevant.» (BBL 2015)

Soll ein externer Dienstleister für Wartung, Support und andere Dienstleistungen für die O-

pen Source Software beauftragt werden, sind die Vorgaben des öffentlichen Beschaffungsrechts zu beachten. Aspekte bezüglich unterschiedlicher Support-Varianten sind in Abschnitt 7 «Support-Modelle beim Einsatz von Open Source Software» beschrieben. Wichtig ist ausserdem der Einsatz passender Eignungs- und Zuschlagskriterien, welche bei Beschaffung von Software-Lösungen bezüglich Open Source Software relevant sind. Die Grundlagen dafür sind in Abschnitt 6 «Analysemöglichkeiten von Open Source Software» erläutert.

3.2 Entwickeln mit Open Source Komponenten

Solange die Software nur für eine bestimmte Organisation weiterentwickelt bzw. betrieben wird, lösen Weitergabehandlungen an einen beauftragten externen Dienstleister (externer Softwareentwickler, Outsourcingpartner), der sich vertraglich zu einer Nichtweitergabe verpflichtet hat, den Copyleft-Effekt grundsätzlich noch nicht aus, da die Handlungen des Dienstleisters der Organisation selbst zugerechnet werden.

In der Literatur wird jedoch kontrovers diskutiert, ob eine interne Weitergabe in einem Konzern bzw. einer öffentlichen Verwaltung generell oder nur innerhalb eines Kreises von eng mit einander verbundenen Personen (z.B. nur innerhalb einer Entwicklungsabteilung) zulässig ist (Analogie zu Art. 9 Abs. 3 URG). Zudem besteht die Gefahr, dass sich weitere Personen (z.B. erst später eingestellte Mitarbeitende) nicht bewusst sind, dass die betreffende Software Komponenten mit Copyleft-Effekt verwendet und sie auch an Dritte weitergeben (z.B. an andere öffentlich-rechtliche Organisationen mit eigener Rechtspersönlichkeit). Daher empfiehlt sich generell, bei der organisationsinternen Weitergabe von modifizierter Software, die einer Lizenz mit Copyleft untersteht, alle Version entsprechend den Vorgaben der betreffenden Open Source Lizenz zu publizieren.

In der modernen Software-Entwicklung wird meist mit Open Source Komponenten gearbeitet, die auf unterschiedliche Art und Weise integriert werden können. Wichtig dabei ist einerseits die Beachtung der jeweiligen Open Source Lizenzen, unter denen die Software-Komponenten veröffentlich sind. Diesbezüglich bestehen insbesondere Vorgaben bei Copyleft-Lizenzen, die stets berücksichtigt werden müssen. In Abschnitt 4 «Eigenschaften von Open Source Lizenzen» werden die wichtigsten rechtlichen Aspekte erläutert. Andererseits spielen die Qualität des Quellcodes und die Zusammensetzung der jeweiligen Community eine wichtige Rolle. Wie diese festgestellt werden können, ist in Abschnitt 6 «Analysemöglichkeiten von Open Source Software» näher erläutert.

3.3 Freigabe von Open Source Quellcode

Bei der Freigabe von Open Source Software muss zwischen dem Beitragen von Quellcode an bestehende Open Source Software und der Veröffentlichung von eigenständigen Open Source Projekten unterschieden werden. Bei Ersterem handelt es sich typischerweise um Fehlerkorrekturen (Bug Fixes) und Erweiterungen von Funktionalitäten (neue Features). Der entsprechende Quellcode muss bzw. kann (abhängig von Lizenztyp und Software-Einsatz) unter der bestehenden Lizenz des Open Source Projekts freigegeben werden. Im zweiten Fall, dem Start eines neuen Open Source Projekts, kann die Lizenz grundsätzlich frei gewählt werden. Einzig muss beachtet werden, unter welcher Lizenz allfällige Open Source Software steht, welche im neuen Open Source Projekt integriert ist. Die Details hierfür sind im Abschnitt 1 «Definitionen» und Abschnitt 4 «Eigenschaften von Open Source Lizenzen» erläutert. Bei vollständigen Neuentwicklungen sollte ein Lizenztyp gewählt werden, welcher eine breite und nachhaltige Basis für Weiterentwicklungen ermöglicht. Dafür ist eine hohe Akzeptanz der betreffenden Lizenz in der entsprechenden Entwickler-Community wichtig.

4. Eigenschaften von Open Source Lizenzen

Im folgenden Abschnitt wird zu Beginn der Begriff des Copyleft erläutert. Anschliessend folgenden die Erläuterungen und Beispiele der drei Arten von Lizenztypen mit starkem, schwachen und keinem Copyleft sowie deren Kompatibilität in Form einer Grafik. Am Ende dieses Abschnitts werden weitere juristische Quellen vorgestellt, welche für vertiefende Aspekte empfohlen sind.

4.1 Copyleft

Eine Open Source Lizenz kann eine sog. «Copyleft»-Bestimmung enthalten. Falls die Bundesverwaltung Software weitergeben möchte, welche OSS mit einer «Copyleft»-Bestimmung enthält, so muss die gesamte Software unter einer Lizenz mit Copyleft veröffentlicht werden. Ist dies der Fall, muss jegliche Veränderung oder Erweiterung des Quellcodes wieder unter der Lizenz der bearbeiteten Open Source Software freigegeben werden. Dies gilt ebenfalls, wenn nur Teile einer Software verwendet werden. Positiv im Sinne der Free Software Foundation ausgedrückt, schützt der Copyleft-Effekt die Freiheit der Entwickler und gewährleistet damit, dass einmal freie Software immer frei bleibt. Negativ formuliert stellt die Copyleft-Bestimmung einen «viralen Effekt» dar, welcher ursprünglich proprietäre Software zur Offenlegung unter einer Open Source Lizenz mit Copyleft erzwingen kann. So ist bei der Verwendung von Copyleft-lizenziertem Quellcode darauf zu achten, dass dieser nur dort integriert wird, wo die resultierende Software wiederum unter einer Open Source Lizenz veröffentlicht wird.

Wenn eine Open Source Lizenz keine Copyleft-Bestimmung enthält (auch permissive Open Source Lizenzen genannt), kann bei einer Bearbeitung der entsprechenden Software die Lizenz frei gewählt werden. So ist es auch möglich, eine proprietäre Software-Lizenz zu verwenden und Open Source Quellcode in proprietäre Software zu integrieren.

Im Wesentlichen können Open Source Lizenzen in drei Kategorien eingeteilt werden:

1. Open Source Lizenzen mit starkem Copyleft,
2. Open Source Lizenzen mit schwachem Copyleft sowie
3. Permissive bzw. Non-Copyleft Open Source Lizenzen.

Auf diese drei Kategorien wird im Folgenden kurz eingegangen. Dabei werden pro Kategorie die bzw. einige der verbreitetsten Open Source Lizenzen aufgeführt.

4.2 Open Source Lizenzen mit starkem Copyleft (Strong Copyleft)

Bei der Verwendung einer Open Source Lizenz mit starkem Copyleft müssen die von der ursprünglichen Software abgeleiteten Werke unter den Bedingungen der Ursprungslizenz stehen. Die folgenden zwei Lizenzen gelten als die relevanten Open Source Lizenzen mit starkem Copyleft:

- GNU General Public License (GPL)
- GNU Affero General Public License (AGPL)

Der wesentliche Unterschied von GPL und AGPL Software bezieht sich auf deren Nutzungsart: Bei GPL-Software muss der veränderte Quellcode nicht mitgeliefert werden, solange dieser bloss auf einer Website (bspw. als Software-as-a-Service Cloud-Dienst) oder über eine Programmier-Schnittstelle (bspw. Application Programming Interface API) zur Verfügung gestellt und kein ausführbares Programm ausgeliefert wird (bspw. Binary File oder Mobile App).

Wenn die Software unter AGPL steht, dann muss der veränderte Quellcode auch dann freigegeben werden, wenn bloss die Funktionalität der Software über eine Web-Oberfläche (bspw. als Software-as-a-Service oder Application Service Providing ASP) zur Verfügung gestellt wird, ohne dass eine ausführbare Software ausgeliefert wird.

4.3 Open Source Lizenzen mit schwachem Copyleft (Weak Copyleft)

Lizenzen mit schwachem Copyleft verlangen, dass Veränderungen an deren Quellcode wiederum unter der ursprünglichen Open Source Lizenz freigegeben werden. Im Gegensatz zu Lizenzen mit starkem Copyleft haben sie jedoch nicht den ‘viralen’ Effekt, dass sie auch weitere abgrenzbare Software-Bestandteile mit ihrer Lizenz ‘anstecken’. Damit wird die Integration von Open Source Software mit schwachem Copyleft in proprietäre Software ermöglicht, ohne dass diese freigegeben werden muss.

Dies sind die meist verwendeten Open Source Lizenzen mit schwachem Copyleft:

- GNU Lesser General Public License (LGPL)
- Mozilla Public License 2.0 (MPL)
- Common Development and Distribution License (CDDL)
- Eclipse Public License (EPL)
- European Union Public License (EUPL)
- Microsoft Reciprocal License (Ms-RL)

4.4 Permissive Open Source Lizenzen (No Copyleft)

Lizenzen ohne Copyleft-Effekt zeichnen sich dadurch aus, dass sie dem Lizenznehmer keine Vorgaben hinsichtlich der Lizenzierung seiner abgeleiteten Software machen und dadurch sowohl dessen neuer Quellcode wie auch Veränderungen an der Open Source Software nicht offengelegt werden muss. So wird die Entwicklung von proprietären Softwareprodukten durch Integration von Software unter permissiven Open Source Lizenzen ermöglicht.

- MIT License
- Apache License 2.0
- Berkley Software Distribution (BSD) License (BSD-2-Clause und BSD-3-Clause)
- Microsoft Public License (Ms-PL)

4.5 Kompatibilität von Open Source Lizenzen

Programme bestehen meist aus einer Vielzahl von Softwarekomponenten und Modulen, welche auf unterschiedliche Weise mit einander verbunden werden können. In der Software-Entwicklung werden bestehende Open Source Komponenten typischerweise häufig in eigene, interne Applikationen und Lösungen integriert. Sobald diese über eine Website, über Programmier-Schnittstellen, über Software-Verteilung oder in einer anderen Form von aussen zugänglich wird, muss auf die Kompatibilität der jeweiligen Open Source Lizenzen geachtet werden. Die nachfolgende Grafik stellt die Abhängigkeiten der wichtigsten Open Source Lizenzen (Goldstein 2018) dar. Die Darstellung basiert auf der Visualisierung von David A. Wheeler, der die Kompatibilitäten der verschiedenen Lizenzen und ihrer Versionsnummern analysiert hat (Wheeler 2007).

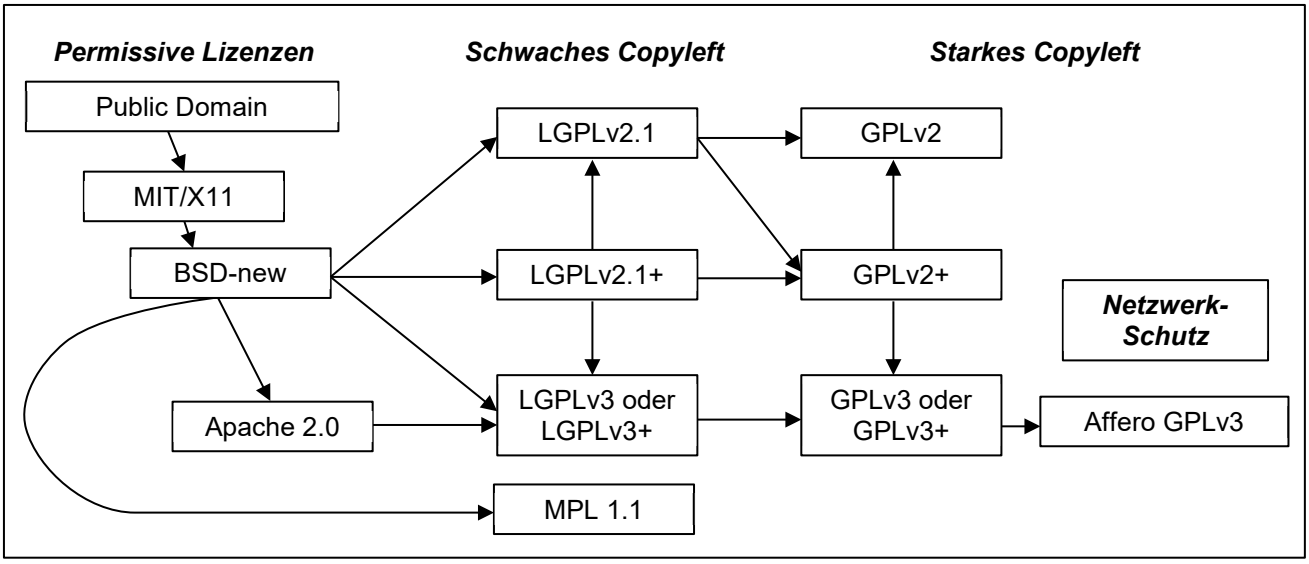

Abbildung 1: Kompatibilität von Open Source Lizenzen (basierend auf Darstellung von David A. Wheeler, 2007)

4.6 Weiterführende Informationen zu rechtlichen Fragestellungen

Weiterführende Informationen zu Copyleft, konkreten Lizenzeigenschaften und vertiefende rechtliche Aspekte können in zahlreichen Publikationen gefunden werden, die nachfolgend vorgestellt werden (siehe dazu auch die Referenzen am Schluss des Dokuments).

- Der «Leitfaden Open-Source-Software 2.0» des deutschen BITKOM geht ausführlich auf Rechtsfragen zu Open Source ein (BITKOM 2016). Er geht allerdings von der Rechtslage in Deutschland aus.
- Eine Zusammenfassung der wichtigsten Open Source Lizenzen sowie eine Kompatibilitätsmatrix wurden in einer Publikation von Ernst & Young veröffentlicht (EY 2011).
- Wolfgang Straub in seinem Buch «Softwareschutz: Urheberrecht, Patentrecht, Open Source» die juristischen Details des Copyleft im Zusammenhang mit dem Schweizer Urheberrecht und vertieft die Kompatibilität von Open Source Lizenzen (Straub 2011). Der Teil über Open Source Software sowie deutsche Übersetzungen verschiedener Open Source Lizenzen unter Berücksichtigung der schweizerischen Rechtsterminologie sind frei verfügbar unter *www.it-recht.ch*
- Vertiefte Antworten zu zahlreichen Rechtsfragen im Zusammenhang mit Open Source Software geben Till Jaeger und Axel Metzger in ihrem umfassenden Buch «Open Source Software – Rechtliche Rahmenbedingungen der Freien Software» (Jaeger und Metzger 2016). Sie gehen allerdings von der Rechtslage in Deutschland aus.

Des Weiteren geben verschiedene Online-Portale detailliert Auskunft zu den Eigenschaften und Fragestellungen von bestimmten Open Source Lizenzen.

- Auf https://choosealicense.com, einer Plattform von GitHub, können die gewünschten Ziele für ein Open Source Projekt ausgewählt werden, worauf die passende Open Source Lizenz vorgeschlagen wird.
- Auf https://opensource.guide/de/legal/ hat GitHub ein Online-Guide zur Verfügung gestellt, der auf konkrete rechtliche Fragen eingeht.

- Auf https://opensource.org/faq geht die Open Source Initiative auf zahlreiche Fragen und Antworten zu rechtlichen Aspekten von Open Source Lizenzen ein.
- Auf https://copyleft.org/guide ist ein ausführlicher Leitfaden publiziert, der die Details des Copyleft erläutert.
- Auf https://tldrlegal.com sind die wichtigsten Open Source Lizenzen zusammengefasst nach Vorgaben, was die jeweilige Lizenz erlaubt («can»), was sie verbietet («cannot») und was sie vorschreibt («must»).
- Auf https://opensource.com/tags/licensing werden fortlaufend Beiträge zu aktuellen Lizenzfragen veröffentlicht.
- Auf https://www.ifross.org/faq-haeufig-gestellte-fragen hat das private «Institut für Rechtsfragen der Freien und Open Source Software» in Berlin (ifrOSS) zahlreiche Antworten auf häufige Rechtsfragen publiziert.

Compliance mit Open Source Lizenzen wird heute vorwiegend mit Software-Tools umgesetzt (siehe zu weiteren Fragen der Open Source Compliance auch Fröhlich-Bleuler 2012 und Kuhn, Williamson, und Sandler 2008). Einerseits gibt es dazu verschiedene Open Source Tools der Linux Foundation unter dem Begriff «fossology», die unter https://www.fossology.org dokumentiert und veröffentlicht sind. Andererseits bieten kommerzielle Anbieter wie Black Duck[7] oder FOSSA[8] verschiedene proprietäre Lösungen an, mit denen die Kompatibilität der eingesetzten Open Source Lizenzen geprüft werden kann.

5. Geschäftsmodelle mit Open Source Software

Open Source Software per se ist kein Geschäftsmodell, da im Gegensatz zu proprietärer Software der Verkauf von Software unter einer Open Source Lizenz keine Option darstellt. Dennoch bestehen für Unternehmen unterschiedliche Möglichkeiten, Geschäftsmodelle basierend auf Open Source Software zu betreiben. Denn wenn beispielsweise für eine Open Source Lösung professioneller externer Support bezogen werden soll, dann ist die Beschaffung von entsprechenden kommerziellen Dienstleistungen notwendig. Im Folgenden sind die vier häufigsten Geschäftsmodelle mit Open Source Software erläutert. Zusätzliche Aspekte dieser und weiterer Geschäftsmodelle sind im BITKOM Leitfaden (BITKOM 2016) ausgeführt.

5.1 Services und Produkte basierend auf Open Source Software

Unternehmen können basierend auf Open Source Software kommerzielle Services wie Web Hosting oder Cloud Computing anbieten, welche mit proprietärer Software wesentlich teurer ausfallen würden. Entsprechend bauen heute die meisten Startups, Online-Portale und E-Commerce Anbieter ihre Plattformen auf Open Source Software auf. Auch andere Technologiefirmen wie Telekomunternehmen, Streaming-Anbieter oder auch Hersteller proprietärer Software integrieren Open Source Software in ihre Software- und Hardware Produkte sowie Online-Services. So können die Unternehmen fortlaufend innovative Lösungen anbieten, die mit proprietärer Software kaum möglich wären.

[7] https://www.blackducksoftware.com

[8] https://fossa.com

5.2 Services für Open Source Software

Explizit Dienstleistungen für ausgewählte Open Source Software erbringen sogenannte Open Source Anbieter. Diese verfügen über erfahrene Open Source Entwickler und können dadurch Support, Wartung, Betrieb, Weiterentwicklung, Beratung, Schulung und weitere Dienstleistungen für Open Source Software anbieten. Diese können beim Open Source Anbieter im Auftragsverhältnis oder Werkvertragsverhältnis bezogen werden. Solche Services für Open Source Software können öffentlich ausgeschrieben werden da keine Abhängigkeit zum jeweiligen Dienstleister (Vendor Lock-In) besteht. Wichtig bei solchen Ausschreibungen ist die Berücksichtigung von entsprechenden Kriterien, damit die tatsächlich kompetenten Dienstleister ausgewählt werden (siehe Abschnitt 6 «Analysemöglichkeiten von Open Source Software»).

5.3 Subscriptions

Werden Services für Open Source Software in einer standardisierten, wiederkehrenden Form als eine Art «Service Level Agreements» (SLA) erbracht, werden diese als «Subscriptions» bezeichnet. Im Rahmen von solchen Subscriptions garantieren die Unternehmen beispielsweise fortlaufende Sicherheits-Updates, Support per Email oder Telefon, Kompatibilität mit anderen Software- und Hardware-Produkten über Zertifizierungen, langfristige Wartungsleistungen und Absicherungen gegen Rechtsansprüche (Copyright, Patente). Im Gegenzug bezahlen Kunden Subscriptions pro Arbeitsplatz oder CPU, ähnlich wie Lizenzgebühren oder Nutzungsgebühren von proprietärer Software. Anders als bei proprietärer Software stellen Subscriptions für Open Source Software jedoch keine Voraussetzung für die Nutzung von Software dar, sondern begleichen lediglich den Mehrwert der jeweiligen Leistungen des Open Source Anbieters.

5.4 Dual Licensing

Wenn ein Unternehmen über das geistige Eigentum einer Software-Lösung verfügt und alle integrierten Open Source Komponenten unter einer permissiven Lizenz stehen, dann kann eine duale Lizenzierung oder auch Mehrfachlizenzierung angewendet werden. Diese erlaubt es der Entwicklerfirma, die Software einerseits unter einer Copyleft Open Source Lizenz zu veröffentlichen und andererseits unter einer proprietären Lizenz zu verkaufen. Diese kommerzielle Variante wird oftmals auch als «Enterprise» Version bezeichnet und beinhaltet typischerweise gewisse zusätzliche Funktionalitäten wie exklusive Schnittstellenintegrationen oder die Erlaubnis an den Käufer, die Software in eigene proprietäre Produkte zu integrieren. Je nach Ausprägung der Einschränkungen der Open Source Variante ist beim Einsatz von dual lizenzierter Software Vorsicht geboten, da der Bezug der Enterprise-Version unumgänglich sein kann und damit die Herstellerabhängigkeit wie bei üblicher proprietärer Software hoch ist.

6. Analysemöglichkeiten von Open Source Software

Verschiedene Plattformen ermöglichen das Auffinden und analysieren von Open Source Software. Die daraus resultierenden Daten stellen eine faktenbasierte Grundlage für Entscheidungen bezüglich Qualität und letztlich Beschaffung von Open Source Software dar.

6.1 alternativeTo

Ein praktisches Tool um Alternativen zu einer Software ausfindig zu machen ist "alternativeTo – Crowdsourced software recommendations".[9] Wie es der Name schon sagt, sind die Alternativen und die Bewertungen dieser Alternativen durch crowdsourcing ("crowd" steht für "Menschenmenge" und "sourcing" für "Beschaffung") durch den Input vieler einzelner Anwender entstanden. alternativeTo unterscheidet vier Arten von Software-Lösungen:

Free Open Source	Software, die unter einer Open Source Lizenz veröffentlicht ist.
Free	Kostenlose Software (Freeware), deren Quellcode jedoch nicht frei zugänglich ist und die nicht verändert werden darf
Freemium	Von solcher Software wird eine kostenlose und eine Premium Version angeboten, wobei bereits in der kostenlosen Version alle wichtigen Funktionalitäten ausgeübt werden können. Nicht zu dieser Kategorie zählt eine Software, die für eine gewisse Zeit gratis getestet werden kann (Bsp. 30-tägige Probeversion), diese fällt in die Kategorie "Commercial".
Commercial	Dabei handelt es sich um proprietäre Software, bei deren Nutzung Lizenzkosten anfallen.

6.2 GitHub

Die weltweit populärste Open Source Entwicklungsplattform mit über 30 Millionen Benutzer und über 100 Millionen Quellcode-Verzeichnissen[10] ist zurzeit GitHub. Heute verfügen praktisch alle Informatik-Unternehmen, insbesondere auch Hersteller proprietärer Software, sowie jegliche andere Art von Firmen und Organisationen über einen GitHub-Auftritt. Dort veröffentlichen die Unternehmen Open Source Applikationen, Tools, Programmier-Bibliotheken etc. Auch immer mehr Behörden – rund ein Dutzend aus der Schweiz[11] – veröffentlichen auf «GitHub and Government» unter https://government.github.com ihre eigene Open Source Software.

Auf GitHub «Insights» lassen sich zahlreiche relevante Statistiken eines Open Source Projekts auf GitHub ablesen:

Pulse	Übersicht der jüngsten Aktivitäten eines Open Source Projekts auf GitHub: Zusammenfassung der wichtigsten Angaben über Entwicklungs-Intensität, Heterogenität der Community, offenen Meldungen, Verbesserungen (Pull Requests) etc.
Contributors	Darstellung welche Entwickler wann wie aktiv gewesen sind: Dies ist ein wichtiger Indikator, ob alles von einer Person abhängt oder ob eine grössere Community dahinter steht
Commits	Anzeige welche Beiträge zu diesem Open Source Projekt in welchem Zeitraum geleistet wurden
Code Frequency	Visualisierung wie viel Quellcode wann hinzugefügt oder entfernt wurde

[9] https://alternativeto.net/
[10] https://octoverse.github.com
[11] https://government.github.com/community/#switzerland

Dependency graph	Abhängigkeiten des Open Source Projekts von anderer Open Source Software (bspw. relevant bei der Identifizierung von Sicherheitslücken und Updates)
Network	Darstellung wann welcher Entwickler einen Beitrag zu welchem Entwicklungs-Ast geleistet hat
Forks	Liste aller Kopien des Open Source Projekts auf GitHub: Indikator der Popularität und Verbreitung der Open Source Software

6.3 Open Hub

Sollen Informationen zu einer Open Source Lösung gesammelt werden, die nicht unbedingt auf GitHub entwickelt wird, bietet sich Open Hub von der Firma Black Duck[12] an. Von rund einer halben Million Open Source Projekten ist eine Vielzahl an wichtigen Informationen übersichtlich zusammengefasst:

Project Summary	Eine kurze Beschreibung des Open Source Projektes wird wiedergegeben.
In a Nutshell	Die wichtigsten Fakten zu einem Open Source Projekt, wie die Anzahl Commits, Contributors und Codezeilen sowie die meistverwendete Programmiersprache, der Zeitpunkt des ersten Commits und der letzten Änderung werden aufgeführt. Zudem wird eine Einschätzung zur Codebasis und der Grösse des Entwicklerteams aufgeführt (Bspw. "Mozilla Firefox has a well established, mature codebase maintained by a very large development team with stable Y-O-Y commits.")
Quick Reference	Beinhaltet Informationen zur Organisation, Links zum Projekt und zum Code sowie Verweise auf ähnliche Projekte.
Licenses	Zeigt an, unter welcher Lizenz bzw. welchen Lizenzen das Open Source Projekt steht und welche Konsequenzen damit verbunden sind.
Project Security	Gibt Auskunft zur Sicherheit und Vulnerabilität des Open Source Projektes.
Code	Die Anzahl Codezeilen im Zeitverlauf werden in einer Grafik visualisiert. Unterschieden werden die verschiedenen Programmiersprachen.
Activity	Die Anzahl Commits pro Monat wird in einer Grafik dargestellt. Zudem wird eine Zusammenfassung der letzten 30 Tage und 12 Monate aufgeführt.
Community	Die Anzahl aktiver Contributors pro Monat wird in einer Grafik dargestellt. Zudem wird ein Rating des Projektes auf einer Fünf-Sterne-Skala angezeigt.

Darüber hinaus bietet Open Hub die Möglichkeit, verschiedene Open Source Projekte miteinander zu vergleichen.[13] Dadurch wird schnell ein Überblick geschaffen, welches Projekt über die aktivste Community, die längste Weiterentwicklungszeit oder die passende Lizenz verfügt.

[12] https://www.openhub.net/
[13] https://www.openhub.net/p/_compare

7. Support-Modelle beim Einsatz von Open Source Software

Der Einsatz von bereits auf dem Markt vorhandener Open Source Software kann grundsätzlich auf drei Arten erfolgen:

1. Einsatz ohne professionellen Support,
2. Einsatz mit internem Support oder
3. Einsatz mit Support durch externen Anbieter.

Diese drei Einsatzarten sowie deren Vor- und Nachteile werden im Folgenden kurz erläutert. Welches dieser Szenarien im spezifischen Fall am sinnvollsten ist, muss situativ entschieden werden. Das Support-Modell wird abhängig von der strategischen Relevanz der Open Source Software, von der technischen Einbindung und der vorhandenen Personal-Ressourcen gewählt.

7.1 Einsatz ohne professionellen Support

Open Source Software wird kostenlos aus dem Internet heruntergeladen, installiert und so wie sie ist eingesetzt.

Vorteile	- Niedrige Kosten - Rasche Umsetzung
Nachteile	- Kein garantierter Support - Keine Haftungsansprüche
Risiko und Absicherung	Hohes Risiko: Es bestehen keinerlei Supportverträge oder Garantien und es existiert wenig bis kein Entwickler-Knowhow bei der Organisation
Typisches Einsatzgebiet	Open Source Standardsoftware, welche unabhängig von anderen Systemen aktualisiert werden kann

7.2 Einsatz mit internem Support

Ein Unternehmen beziehungsweise eine öffentliche Institution baut intensiv Knowhow und Ressourcen zu bestimmten Open Source Lösungen auf um diese intensiv und langfristig einzusetzen. Dieses Vorgehen wird insbesondere auch in geschäftskritischen Bereichen angewendet.

Vorteile	- Hohe Flexibilität dank internem Know-how - Keine Anbieterabhängigkeiten
Nachteile	- Hohe Investitionen und grosser Zeitaufwand durch Know-how Aufbau - Höhere interne Fixkosten für Mitarbeitende
Risiko und Absicherung	Mittleres Risiko: Der Support hängt von Knowhow und Verfügbarkeit der internen IT ab
Typisches Einsatzgebiet	Inhouse-Entwicklungen, strategisch eingesetzte Open Source Software zu der vertieftes Knowhow vorhanden ist

7.3 Einsatz mit Support durch externen Anbieter

Ein externer Open Source Anbieter wird beigezogen, der die Einführung und Wartung der Open Source Software professionell begleitet. Dieses Vorgehen wird insbesondere auch in geschäftskritischen Bereichen angewendet in denen unmittelbar vertieftes Know-how der Software vorhanden sein muss.

Vorteile	- Direkter Zugang zum Know-how der Open Source Entwickler - Korrekturen und Weiterentwicklung auf Auftragsbasis - Auswahl verschiedener Open Source Anbieter - Verbindlichkeit, Absicherung von Compliance-Risiken
Nachteile	- Externe Kosten durch Open Source Anbieter - Know-how-Abhängigkeit zum Open Source Anbieter
Risiko und Absicherung	Niedriges Risiko: Gewährleistung geschieht gemäss Auftragsbeschreibung oder Service Level Agreement
Typisches Einsatzgebiet	Geschäftskritische Open Source Software, zu der kein oder wenig internes Entwickler-Knowhow vorhanden ist

8. Freigabe von Open Source Software

Um die rechtliche Konformität sicherzustellen, damit die Bundesverwaltung OSS freigeben und in Communities mitarbeiten darf, wurde ein externes Rechtsgutachten erstellt. Dieses kam zum Schluss, dass die dafür notwendigen Gesetzesgrundlagen fehlen würden. Kurze Zeit später liess der Kanton Bern für sich ein Gutachten erarbeiten, welches ihm die Rechtsmässigkeit der gleichen Tätigkeiten zusicherte.

Um Rechtssicherheit zu schaffen, sollen die Freigabe von OSS und die Mitarbeit in Communities nun im «Bundesgesetz über die Digitale Verwaltung» verankert werden. Dieses Gesetz soll Anfang 2020 in Vernehmlassung gehen.

In der Zwischenzeit liegt es in der Verantwortung der einzelnen Dienststellen, ob diese OSS freigeben und in Communities mitarbeiten wollen oder nicht. Die Risikoabwägung muss dabei für jeden einzelnen Fall erfolgen.

9. Fragen aus der Praxis

Nachfolgend werden Antworten auf einige typische Fragen aus der Praxis gegeben.

1. **Was sind die Voraussetzungen / Rahmenbedingungen um Rechtssicherheit beim Austausch von Source Code zu haben?**

   Die OSS Komponenten können beliebig mit Drittkomponenten kombiniert werden und es besteht keine Publikationspflicht solange die Software ausschliesslich innerhalb der gleichen Organisation genutzt und nicht weitergegeben wird. Falls ein eine Weitergabe über die Grenzen der Organisation hinaus stattfindet, dann müssen die entsprechenden OSS-Lizenz-Bedingungen eingehalten werden. Das heisst bspw. bei GPL-Quellcode, dass alle Weiterentwicklungen der betroffenen Software ebenfalls unter der GPL freigegeben werden müssen (siehe dazu auch Abschnitt 3 «Konstellationen der Nutzung von Open Source Software»).

2. Unter welcher Open Source Lizenz soll Software entwickelt werden?

Es gibt keine Vorgaben hinsichtlich der Lizenzwahl, solange die Lizenzbestimmungen der Software-Komponenten eingehalten werden, welche in der zu entwickelnden Software verwendet werden (siehe dazu insbesondere Abschnitt 3 «Konstellationen der Nutzung von Open Source Software» und 4.1 «Copyleft»). Bei vollständigen Neuentwicklungen sollte ein Lizenztyp gewählt werden, welcher eine breite und nachhaltige Basis für Weiterentwicklungen ermöglicht. Dafür ist eine hohe Akzeptanz der betreffenden Lizenz in der entsprechenden Entwickler-Community wichtig.

3. Spielt es eine Rolle ob OSS kommerziell oder nicht kommerziell benutzt wird?

Open Source Lizenzen unterscheiden grundsätzlich nicht zwischen kommerzieller und nicht kommerzieller Nutzung. Deshalb kann Open Source Software für beliebige Zwecke eingesetzt werden, auch für kommerzielle Anwendungen. Kommerzielle Anbieter versuchen oft OSS Komponenten in proprietäre Produkte zu integrieren. Dies ist nur zulässig, wenn die OSS Komponenten nicht unter einer Lizenz mit Copyleft-Effekt stehen (siehe dazu insbesondere Abschnitt 4.1 «Copyleft»).

4. Unter welcher Autorität darf ein Bundesamt eigene Software unter einer Open Source Lizenz weitergeben?

Diese Frage hängt vom Entscheid des Bundesrats bezüglich der Freigabe von Open Source Software ab (siehe «Freigabe von Open Source Software»). Im Kanton Bern können Behörden seit 2018 eigene Software unter einer Open Source Lizenz freigeben (Schlauri, Schweizer, und Poledna 2017).

5. Wie sollen Eigenentwicklungen eines Bundesamtes publiziert werden, die in Kooperationen entstanden sind? Wie soll mit bestehenden Rechten umgegangen werden?

Das Urheberrecht entsteht bei den jeweiligen natürlichen Personen, die den konkreten Quellcode entwickelt haben, wobei diese meist arbeitsvertraglich an den Arbeitgeber abgetreten werden. Bei Verträgen mit externen Leistungserbringern ist wichtig, dass eine klare Abgrenzung der Neuentwicklungen stattfindet und eine entsprechende vertragliche Zuordnung/Übertragung des Urheberrechts vereinbart wird. Letztlich müssen sich bei gemeinsamen Eigenentwicklungen die beteiligten Parteien über die Open Source Lizenz des Quellcodes einigen, unter der Software als eigenständiges Open Source Projekt publiziert wird. Die Publikation unter einer Open Source Lizenz bedeutet keinen Verzicht auf das Urheberrecht. Vielmehr bleibt dieses bei den jeweiligen Rechtsinhabern. Nur aufgrund ihres Urheberrechts können sie sich rechtlich gegen Lizenzverletzungen wehren oder parallel zu OSS Lizenzen noch weitergehende Lizenzen erteilen.

6. Wie werden Guidelines für die Entwicklung geregelt?

Es bestehen nebst den allgemeinen Vorgaben keine speziellen Vorgaben bezüglich Open Source Software.

7. Was sind für Vorgaben an Quality Gates geplant?

Es bestehen nebst den allgemeinen Vorgaben keine speziellen Vorgaben bezüglich Qualität von OSS-Software.

8. Was sind die rechtlichen Vorgaben für die Weitergabe von Quellcode?

Es müssen sowohl die Lizenz-rechtlichen Aspekte wie Copyleft (siehe Abschnitt 4.1 «Copyleft») als auch die Bund-internen – derzeit noch zu erstellenden – Weisungen bezüglich Freigabe von Open Source Software berücksichtigt werden.

9. Müssen Veränderungen veröffentlicht werden?

Nur Softwareversionen, welche weitergegeben werden, müssen veröffentlicht werden.

10. Was ist eine genügende Veröffentlichung?

Die anwendbare OSS Lizenz bestimmt die Modalitäten wie der Sourcecode gegenüber Dritten verfügbar gemacht werden muss.

11. In welchem Kreis kann OSS weitergeben werden, ohne dass Copyleft greift? Was gilt innerhalb der Bundesverwaltung als ‚Organisation' bzw. ‚Unternehmen'?

Soweit ersichtlich gibt es in der Schweiz bisher keine Rechtsprechung zu dieser Frage. In der Literatur wird jedoch kontrovers diskutiert, ob eine interne Weitergabe in einem Konzern bzw. einer öffentlichen Verwaltung generell oder nur innerhalb eines Kreises von eng mit einander verbundenen Personen (z.B. nur innerhalb einer Entwicklungsabteilung) zulässig ist (Analogie zu Art. 9 Abs. 3 URG). Zudem besteht die Gefahr, dass sich weitere Personen (z.B. erst später eingestellte Mitarbeitende) nicht bewusst sind, dass die betreffende Software Komponenten mit Copyleft-Effekt verwendet und sie auch an Dritte weitergeben (z.B. an andere öffentlich-rechtliche Organisationen mit eigener Rechtspersönlichkeit). Daher empfiehlt sich generell, bei der organisationsinternen Weitergabe von modifizierter Software, die einer Lizenz mit Copyleft untersteht, alle Version entsprechend den Vorgaben der betreffenden Open Source Lizenz zu publizieren.

10. Weiterentwicklung des Praxisleitfadens

Dieser Praxisleitfaden soll kontinuierlich weiterentwickelt werden. Die jeweils aktuelle Version befindet sich immer auf der Webseite des ISB.

Referenzen

BBL. 2015. *Merkblatt Software-Ausschreibungen: Sicherstellung eines breiten Wettbewerbs*. Kompetenzzentrum Beschaffungswesen Bund (KBB) und Rechtsdienst Bundesamt für Bauten und Logistik (BBL). https://www.beschaffung.admin.ch/dam/bpl/de/dokumente/Bedarfsstellen/Merkblaetter/Merkblatt_Software_Ausschreibungen_inkl_%20IT_ABG_und_Pflichtenheftbeilage.pdf.download.pdf/Merkblatt_Software_Ausschreibungen_inkl_%20IT_ABG_und_Pflichtenheftbeilage.pdf.

BITKOM. 2016. *Leitfaden Open-Source-Software 2.0*. Berlin: Bitkom e. V. Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e. V. https://www.bitkom.org/Bitkom/Publikationen/Bitkom-Leitfaden-zu-Open-Source-Software-20.html.

EY. 2011. *Open Source Software im geschäftskritischen Einsatz*. Ernst & Young. https://www.ossdirectory.com/oss-knowhow/details/kbarticle/open-source-software-im-geschaeftskritischen-einsatz/.

Fröhlich-Bleuler, Gianni. 2012. «Open Source Compliance». *Jusletter* 12. November 2012: 10.

Goldstein, Ayala. 2018. «Top 10 Open Source Licenses in 2018: Trends and Predictions». https://resources.whitesourcesoftware.com/blog-whitesource/top-open-source-licenses-trends-and-predictions (2. Juni 2019).

Jaeger, Till, und Axel Metzger. 2016. *Open Source Software: Rechtliche Rahmenbedingungen der Freien Software*. 4. Aufl. München: C.H.Beck.

Kuhn, Bradley M., Aaron Williamson, und Karen M. Sandler. 2008. «A Practical Guide to GPL Compliance». *Software Freedom Law Center*. https://softwarefreedom.org/resources/2008/compliance-guide.html (26. August 2019).

Open Source Initiative. 2019. «Open Source Licenses by Name». https://opensource.org/licenses/alphabetical.

Perens, Bruce. 1999. «The Open Source Definition». In *Open Sources: Voices from the Open Source Revolution*, https://www.oreilly.com/openbook/opensources/book/perens.html.

Schlauri, Simon, Samuel Schweizer, und Tomas Poledna. 2017. *Rechtliche Voraussetzungen Der Nutzung von Open-Source-Software in Der Öffentlichen Verwaltung, Insbesondere Des Kantons Bern*. Carl Grossmann Verlag. http://www.oapen.org/search?identifier=632680 (26. August 2019).

Straub, Wolfgang. 2011. *Softwareschutz: Urheberrecht, Patentrecht, Open Source*. Zürich: Dike. https://www.it-recht.ch/wp-content/uploads/2014/11/Straub-Softwareschutz-Open-Source-Software-Zurich-2011.pdf.

Stürmer, Matthias, und Carole Gauch. 2018. *Open Source Studie Schweiz 2018*. Forschungsstelle Digitale Nachhaltigkeit der Universität Bern. https://oss-studie.ch.

Wheeler, David A. 2007. *The Free-Libre / Open Source Software (FLOSS) License Slide*. https://dwheeler.com/essays/floss-license-slide.pdf.