NUTZUNG VON OPEN-SOURCE-SOFTWARE IM DLR

Version 2.0

INHALTSVERZEICHNIS

Sprachlich orientiert sich die Broschüre an den Maßgaben des „Leitfaden – Gendergerechte Sprache im DLR“. Der Text soll sachlich korrekt, verständlich und (vor)lesbar sein sowie Rechtssicherheit und Eindeutigkeit gewährleisten. Dabei sind selbstverständlich stets Personen jeden Geschlechts gleichermaßen angesprochen.


print(“In real open source, you have the right to control your own destiny.”)
Linus Torvalds

1 Vorwort

Open-Source-Software ist ein integraler Bestandteil moderner Softwareentwicklung. Das Deutsche Zentrum für Luft- und Raumfahrt e. V. (DLR) greift bei der Entwicklung von Software vielfach auf Open-Source-Software zurück und stellt gleichzeitig seinerseits dem Markt vielfach Open-Source-Software bereit. Zudem tragen einzelne DLR-Softwareentwickelnde immer wieder zu bestehenden Open-Source-Software-Projekten Dritter bei.

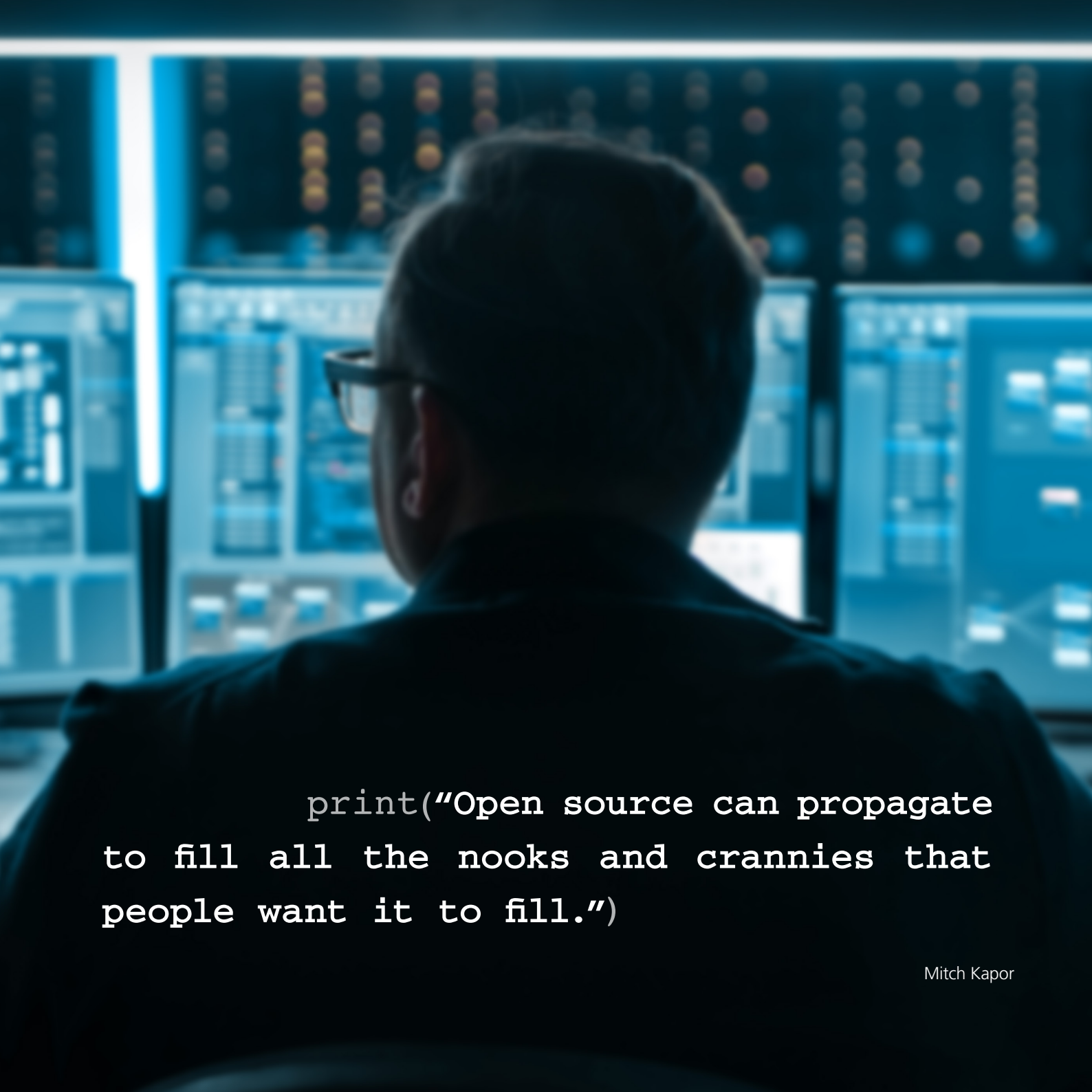
In diesem Zusammenhang stellen sich häufig technische und praktische Anwenderfragen sowie rechtliche Lizenzfragen. Im April 2013 hat der Bereich DLR-Technologietransfer gemeinsam mit dem Institut für Softwaretechnologie und mit Unterstützung der Kanzlei JBB Rechtsanwälte die erste Broschüre zur Nutzung von Open-Source-Software im DLR veröffentlicht. Diese Broschüre hat weit über das DLR hinaus, unter anderem in der gesamten Helmholtz-Gemeinschaft, großen Anklang gefunden und ist in der gedruckten Version seit Langem vergriffen.

Daher, und auch, um den Veränderungen der letzten Jahre Rechnung zu tragen, stellen wir heute eine grundlegend aktualisierte Fassung der Broschüre zur Nutzung von Open-Source-Software im DLR vor. Diese Broschüre soll zum einen noch anwenderfreundlicher sein als die Vorgängerbroschüre. Zum anderen enthält sie erstmals einen Abschnitt, in dem die Position des DLR zu Open-Source-Software dargelegt wird, Hinweise zur Veröffentlichung von Software als Open-Source-Software enthalten sind und Empfehlungen zur Verwendung verschiedener Open-Source-Lizenzen gegeben werden.

Wir wünschen Ihnen viel Spaß bei der Lektüre und hoffen, dass Ihre Fragen rund um das Thema Open-Source-Software beantwortet werden.

Bernhard Milow
Leiter DLR-Technologietransfer

Rolf Hempel
Leiter DLR-Institut für Softwaretechnologie



print(“Open source can propagate to fill all the nooks and crannies that people want it to fill.”)

Mitch Kapor

2 Eckpunkte der DLR Open-Source-Software-Politik

Software ist eine digitale Währung der Zukunft. Für eine mit öffentlichen Mitteln finanzierte Forschungseinrichtung wie das DLR ist es wichtig, seine Leistung der Öffentlichkeit und insbesondere dem Markt zugänglich zu machen und so seiner satzungsgemäßen Aufgabe nachzukommen. Hierfür braucht es einen gestalteten Technologietransfer, denn die Ergebnisse der Forschung sind vielfältig. Es entstehen unter anderem Erfindungen, Patente, Gebrauchsmuster, Designs, Know-how und urheberrechtlich geschützte Werke, wie z. B. Software. Alle diese Ergebnisse zählen zum Bereich des geistigen Eigentums und tragen dazu bei, die wissenschaftliche Leistung einer Forschungseinrichtung zu verdeutlichen.

Dabei steht hervorgehoben die Möglichkeit einer kommerziellen Verwertung des Forschungserfolgs im Fokus, schon deswegen, um der öffentlichen Finanzierung der Ergebnisse Rechnung zu tragen und der Öffentlichkeit einen monetären Gegenwert für ihre Finanzierungsbereitschaft zu erhalten. Im Falle von Software geschieht dies in Form von individuell gestalteten Softwarelizenzverträgen, die für die Einräumung von Nutzungsrechten an der Software die Zahlung einer marktüblichen Lizenzgebühr vorsehen. Dabei werden von Fall zu Fall die Art und der Umfang der Lizenz betrachtet, um zu ausgewogenen und diskriminierungsfreien Vereinbarungen zu gelangen, die das lizenzierte Ergebnis weiterhin einer möglichst breiten Interessengruppe verfügbar halten sowie die weitere Forschung des DLR am Thema ermöglichen.

Darüber hinaus kann es in etlichen Fällen sinnvoll sein, der Öffentlichkeit eine Software unentgeltlich als Open-Source-Software zur Verfügung zu stellen, z. B. um die Reproduzierbarkeit von darauf aufbauenden Forschungsergebnissen sicherzustellen. Ein anderes Beispiel ist der Fall, in welchem ein kommerzielles Verwertungsfeld zunächst nicht gegeben ist. Hier kann durch die Zurverfügungstellung als Open-Source-Software eine besondere Marktdurchsetzung zur Schaffung eines Standards erzielt werden, auf dem zukünftig mit eigener, kommerziell zu lizenzierender Software aufgebaut werden kann.

Die große und stetig größer werdende Bedeutung von Open-Source-Software als integralem Bestandteil moderner Softwareentwicklung macht eine organisationspolitische Positionierung hierzu sinnvoll und erforderlich. Dies erfolgt entlang der Fragen: Wie steht das DLR zu Open-Source-Software? Was gilt es bei der Veröffentlichung von Software als Open-Source-Software zu beachten und welche Open-Source-Lizenz sollte gewählt werden? Die Antworten auf diese Fragen gibt der nachfolgende Teil der Broschüre.

2.1 Position zu Open-Source-Software

Ein Aspekt für die Bereitstellung und Nutzung von Open-Source-Software ist eine bessere Überprüfbarkeit wissenschaftlicher Ergebnisse, die das Markenzeichen guter wissenschaftlicher Praxis ist. Das DLR lädt dazu ein, die von ihm erstellte Open-Source-Software zu verbreiten und weiterzuentwickeln und es wird seinerseits die eigenen Weiterentwicklungen bestehender Open-Source-Software der Community zur Verfügung stellen. Dies eröffnet den Instituten und Einrichtungen des DLR neue Möglichkeiten der Kooperation sowie gleichzeitig die Möglichkeit, zu Transparenz, Austausch und Diskursen beizutragen.

Allerdings gibt es auch einen Wettbewerb um knappe Drittmittel und Aufträge. Daher ist der Kommerzialisierungsgedanke im Zusammenhang mit Open-Source-Software keineswegs ein Widerspruch. Denn diese Art der Softwareentwicklung eröffnet ein neues Zusammenspiel von Partnern. Open-Source-Software ist freie Software und gibt den Nutzenden der Software als solchen das Recht, diese ablaufen zu lassen, zu vervielfältigen, zu verändern und zu teilen. „Frei" ist dabei zu verstehen im Sinne von „freedom", nicht im Sinne von kostenlos. Es kann (und darf!) mit Open-Source-Software Geld verdient werden. "To understand the concept, you should think of 'free' as in 'free speech', not as in 'free beer'."[1] Diesen Standpunkt vertritt auch das DLR. Denn tatsächlich eröffnen sich mit Open-Source-Software interessante neue – auch kommerzielle – Verwertungsmöglichkeiten. Dies können beispielsweise Service-, Support- und Wartungsdienstleistungen sein, Beratungen und Schulungen oder die Entwicklung kundenspezifischer Funktionalitäten. Letzteres kann im Transfer von Wissenschaft und Forschung hin zu Wirtschaft und Industrie in Form der Auftragsforschung oder im Wege einer klassischen kommerziellen Lizenzierung geschehen. Ein Teil dieser Möglichkeiten wird im DLR bereits genutzt. In der jüngeren Vergangenheit hat sich eine ganze Reihe unterschiedlicher Geschäftsmodelle für nahezu jeden denkbaren Dienst im Zusammenhang mit Open-Source-Software herausgebildet. Einen guten Überblick hierüber gibt der Leitfaden „Open-Source-Software 2.0" des Bitkom[2], des Digitalverbandes Deutschlands, in welchem das DLR Mitglied ist.

Schließlich steht das DLR auch im Wettbewerb um die besten Talente. Durch die Bereitstellung von und Mitarbeit an Open-Source-Software macht das DLR auf die eigenen spannenden Themen aufmerksam und bietet Mitarbeitenden eine attraktive persönliche Weiterentwicklungsmöglichkeit an.
Das DLR unterstützt somit grundsätzlich sowohl die Verwendung von Open-Source-Software bei der Erstellung eigener Software als auch die Weitergabe eigener Software als Open-Source-Software, sofern dies gegenüber der kommerziellen Lizenzierung vorteilhaft ist. Die Entscheidung hierüber trifft die jeweilige Instituts- bzw. Einrichtungsleitung. Gleiches gilt für den Fall, dass DLR-Softwareentwickelnde zu bestehenden Open-Source-Software-Projekten Dritter beitragen.

2.2 Ablauf zur Veröffentlichung von Open-Source-Software im DLR

Häufig wird angenommen, dass es ausreicht, den Quellcode verfügbar zu machen, um eine Software als Open-Source-Software zu veröffentlichen. Dies ist aber nicht der Fall (siehe 3.2). Weitere wesentliche Schritte sind die Auswahl einer Lizenz sowie die Evaluierung und Berücksichtigung etwaiger Rechte Dritter. Im DLR unterliegt die Lizenzierung und Veröffentlichung von Software als Open-Source-Software weiterhin organisatorischen Regelungen. Der im Folgenden beschriebene Prozess deckt beide Bereiche ab. Ziel des Prozesses ist es, Wissenschaftlerinnen und Wissenschaftler bei der Veröffentlichung von Software als Open-Source-Software zu unterstützen, sodass es zum einen nicht zu Rechtsverletzungen Dritter kommt und zum anderen die veröffentlichte Software den Grundlagen von Open-Source-Software entspricht.

[1] https://www.gnu.org/philosophy/free-sw.en.html

[2] Bitkom, Open-Source-Software 2.0, 2016, S. 24 ff., abrufbar unter: https://www.bitkom.org/Bitkom/Publikationen/Bitkom-Leitfaden-zu-Open-Source-Software-20.html

Das DLR bietet Unterstützungsangebote an, welche DLR-Mitarbeitende insbesondere bei der Evaluierung der rechtlichen Rahmenbedingungen unterstützen. Neben dieser Broschüre sind dies Beratungs- und Austauschmöglichkeiten zu den im Folgenden genannten Unterthemen.

Prozess zur Veröffentlichung von Software als Open-Source-Software:

1. Einverständnis der Instituts-/Einrichtungsleitung einholen

Die Instituts-/Einrichtungsleitung (ggf. delegiert) muss der Veröffentlichung der Software als Open-Source-Software zustimmen. In diesem Zusammenhang sollten die Vor- und Nachteile der Veröffentlichung als Open-Source-Software abgewogen werden.

2. Rechtliche Rahmenbedingungen evaluieren

a. Mitbestimmungsrelevante Urheber ermitteln
Sämtliche Personen, die einen relevanten Beitrag zur Software geleistet haben, gelten als mitbestimmungsrelevante Urheber der Software. Damit das DLR eine Software unter einer Open-Source-Lizenz veröffentlichen kann, müssen diese Personen dem zustimmen bzw. die erforderlichen Rechte an das DLR übertragen. Im Fall von mitbestimmungsrelevanten Urhebern mit einem DLR-Arbeitsvertrag ist Zweiteres in der Regel bereits der Fall (siehe 3.1.2.2). Für Externe kann die Rechteübertragung hingegen mittels eines sog. Contributor License Agreements (CLA, siehe 3.2.3) erfolgen. Bei Freelancern und Zuliefernden kann dies durch entsprechende Lizenzverträge geschehen.

b. Exportbeschränkungen der Software klären
Mit der Veröffentlichung einer Software unter einer Open-Source-Lizenz findet gewissermaßen ein weltweiter Export der Software statt. Bestimmte Güter (in Form von Waren, Software etc.) unterliegen jedoch Exportbeschränkungen und können nicht ohne Weiteres exportiert werden. Dies gilt insbesondere für Güter, die (auch) militärisch nutzbar sind.

c. Förder-/Auftrags-/Kooperationsbedingungen prüfen
Die Entwicklung einer Software kann in unterschiedlicher Art und Weise finanziert und gefördert sein. Die Förder-/Auftragsbedingungen können Einschränkungen bzw. Bedingungen hinsichtlich der Veröffentlichung der Software enthalten. Gleiches gilt für Kooperationsverträge mit Kooperationspartnern in Förderprojekten (z. B. Konsortialverträge in EU-Projekten) oder in Kooperationsprojekten, die ohne Drittmittelunterstützung grundfinanziert durchgeführt werden.

d. Lizenzbedingungen von genutztem Code Dritter analysieren
Eine Software baut in der Regel auf Code Dritter (z. B. in Form von Bibliotheken, von Webseiten oder aus Büchern) auf. Auch wenn dieser Code frei zugänglich ist, unterliegt dessen Weiterverbreitung als Teil der eigenen Software regelmäßig bestimmten Lizenzbedingungen. Zudem kann die kombinierte Nutzung verschiedener Bibliotheken durch sich widersprechende Lizenzbedingungen eine Weiterverbreitung rechtlich unmöglich machen (siehe hierzu auch 3.1.4 sowie die praktischen Anwendungsfälle in Kapitel 5 dieser Broschüre).

e. Abschließende Kosten-Nutzen-Betrachtung durchführen und weiteres Vorgehen abstimmen
Auf Basis der bisher gesammelten Informationen lassen sich die anstehenden Aufgaben (z. B. Anpassungen der Software, einzuholende Rechte Dritter etc.) und deren Aufwand abschätzen. Daher sollte zu diesem Zeitpunkt der abgeschätzte Aufwand dem angestrebten Nutzen gegenübergestellt werden. Auf dieser Basis ist mit den relevanten Beteiligten eine Entscheidung über das weitere Vorgehen zu treffen.

3. Veröffentlichungsstrategie festlegen und vorbereiten

a. Open-Source-Lizenz, Veröffentlichungsform und Support-Strategie festlegen
Die Open-Source-Lizenz, unter der die Software veröffentlicht wird, ist festzulegen. Deren Auswahl hängt insbesondere von den Gründen der Veröffentlichung (Nr. 1), einzuhaltenden Förder-/Auftrags-/Kooperationsbedingungen (Nr. 2c) und Lizenzbedingungen von Code Dritter (Nr. 2d) ab. Zudem ist zu überlegen, wo und in welcher Form die Software zugänglich gemacht wird. Schließlich ist festzulegen, wie prinzipiell mit Beiträgen Dritter und Support-Anfragen umgegangen wird (siehe 3.2.3 und 3.2.4).

b. Software für die Veröffentlichung vorbereiten
In Vorbereitung der Veröffentlichung sind die bisher identifizierten Aufgaben zu erledigen und die Software für die Veröffentlichung aufzubereiten (siehe hierzu auch 3.2.1).

c. Voraussetzungen für die Veröffentlichung prüfen und die Software freigeben
Die Instituts-/Einrichtungsleitung (ggf. delegiert) muss die Software zur Veröffentlichung freigeben.

4. Software veröffentlichen

In diesem letzten Schritt wird die Software wie zuvor abgestimmt veröffentlicht.

Organisatorische Aspekte der Schritte können in den Instituten/Einrichtungen unterschiedlich geregelt sein. Mitarbeitende sollten daher Rücksprache halten mit den jeweiligen Ansprechpartnerinnen und Ansprechpartnern (wie zum Beispiel der Ansprechpartnerin oder dem Ansprechpartner zum Thema Software Engineering des Instituts bzw. der Einrichtung).

2.3 Empfehlungen zu Open-Source-Lizenzen

Die Wahl der passenden Open-Source-Lizenz kann mitunter schwierig sein, denn es existieren Dutzende Open-Source-Lizenzen unterschiedlicher Art. Teilweise unterscheiden sich diese lediglich hinsichtlich einzelner Formulierungen, nicht aber inhaltlich. Teilweise bestehen indessen wesentliche strukturelle Unterschiede.

Open-Source-Lizenzen lassen sich in drei Kategorien einteilen: Lizenzen mit starkem Copyleft, Lizenzen mit beschränktem Copyleft und Lizenzen ohne Copyleft, sog. Permissive Lizenzen (zu den einzelnen Lizenzen siehe Kapitel 4). Wichtig ist in jedem Fall, dass man sich für eine allgemein anerkannte Open-Source-Lizenz entscheidet. Nicht anerkannte Open-Source-Lizenzen werden von potenziellen Nutzenden häufig gemieden, da sie individuell evaluiert werden müssen und sich darüber hinaus schwer absehbare Kompatibilitätsprobleme im Zusammenhang mit der Verwendung anderer Open-Source-Software ergeben können. Eine Liste der allgemein anerkannten Open-Source-Lizenzen findet sich auf der Website der Open Source Initiative.[3]

[3] https://opensource.org

Im Übrigen hängt die Wahl der passenden Open-Source-Lizenz entscheidend davon ab, welche Ziele mit der Veröffentlichung der Software als Open-Source-Software verfolgt werden: Soll sich die Software schlicht möglichst weit verbreiten, dann empfiehlt sich eine Permissive Lizenz, wie beispielsweise die Apache-, BSD- oder MIT-Lizenz. Diese Lizenzen lassen den Nutzenden weitgehende Freiheiten und machen die Software somit attraktiv, insbesondere auch für am Markt agierende Wirtschaftsunternehmen. Soll hingegen sichergestellt werden, dass etwaige Weiterentwicklungen der Software ebenfalls allein unter Open-Source-Lizenzbedingungen weitergegeben werden können, muss eine Lizenz mit (starkem) Copyleft gewählt werden, wie die (L)GPL-Lizenzen. In bestimmten Fällen kann sich auch ein sog. Dual Licensing empfehlen. Hierbei wird die Software sowohl unter einer Copyleft-Lizenz angeboten als auch unter einer eigenen kommerziellen Lizenz. Wird die Software von den Nutzenden unter der Copyleft-Lizenz bezogen, ist sichergestellt, dass auch darauf aufbauende Entwicklungen wiederum allein unter der ursprünglichen Copyleft-Lizenz weitergegeben werden können und somit für jeden frei verfügbar sind. Möchten die Nutzenden der Software dies nicht, müssen sie die Software unter der angebotenen kommerziellen Lizenz beziehen. Mit darauf aufbauenden Entwicklungen können sie dann verfahren, wie sie möchten (abhängig von den jeweiligen Bedingungen der kommerziellen Lizenz).

Ferner ist bei der Wahl der passenden Open-Source-Lizenz die Kompatibilität zu anderen Open-Source-Lizenzen zu bedenken. Soll die Software zum Beispiel in ein vorhandenes Software-Element integriert werden, das unter einer GPL-Lizenz steht, muss die gewählte Lizenz zur GPL kompatibel sein. Die Frage der Lizenzkompatibilität stellt sich immer dann, wenn mindestens ein Software-Element unter einer Lizenz mit Copyleft steht (zur Lizenzkompatibilität siehe 3.1.4).

Zuletzt kann die Wahl der Open-Source-Lizenz auch bestimmt sein durch das Projekt, in dem die Software entstanden ist oder in das sie einfließen soll. Häufig ist hier eine bestimmte Lizenz von vornherein vorgegeben. Ein Beispiel ist die Eclipse Foundation, in der das DLR Mitglied ist. Bei der Eclipse Foundation handelt es sich um eine seit 2004 bestehende gemeinnützige Gesellschaft, bestehend aus derzeit über 300 Mitgliedern, deren Ziel die (Weiter-)Entwicklung von Open-Source-Software insbesondere für den wirtschaftlichen Einsatz ist. Als Ergebnis der gemeinsamen Bemühungen wird eine qualitativ hochwertige Software angestrebt, die nicht nur von jedem genutzt, sondern auch rechtssicher innerhalb wirtschaftlicher Vorhaben verwendet werden kann. Software innerhalb der Eclipse Foundation wird zwecks Vereinheitlichung und Standardisierung unter der EPL-2.0 lizenziert, einer Open-Source-Lizenz mit beschränktem Copyleft.

2.4 Empfehlungen zu Veröffentlichungsstrategien

Die Veröffentlichungsstrategie umfasst die Lizenzwahl, die Veröffentlichungsform (siehe 3.2.2), die Einstellung zu Beiträgen Dritter (siehe 3.2.3) sowie die Support-Strategie (siehe 3.2.4). Die Festlegung einer Veröffentlichungsstrategie ist nicht trivial und sollte unbedingt auf das Ziel der Veröffentlichung ausgerichtet sein.

Im DLR typische Ziele für die Veröffentlichung von Software als Open-Source-Software sind:

1. Veröffentlichung begleitend zu einer Publikation
2. Veröffentlichung zur Nutzung durch die Community
3. Veröffentlichung zur Weiterentwicklung durch die Community
4. Veröffentlichung zur Weiterentwicklung mit Partnern

Die genaue Veröffentlichungsstrategie ist projektspezifisch.

Die Software-Engineering-Rahmenrichtlinie des DLR[4] bildet dabei die Basis mit Empfehlungen zur Aufbereitung der Software. In der Richtlinie werden drei Anwendungsklassen von Projekten definiert und entsprechende Empfehlungen bezüglich guter Praxis bei der Softwareentwicklung ausgesprochen. Für Software des DLR empfehlen wir. bezüglich der Veröffentlichungsstrategie weiterhin Folgendes zu berücksichtigen:

- Für Anwendungsfälle, bei denen es vorrangig um die Publikation geht, ist es ausreichend, das Projekt aufzubereiten (siehe 3.2.1), den Quellcode zur Verfügung zu stellen (siehe 3.2.2) und minimalen Support anzubieten (siehe 3.2.4). Dies hält den Aufwand minimal, gewährleistet aber, dass Dritte die Software generell nutzen können.

- Steht die Nutzung der Software durch die Community aber im Vordergrund, so sollten zusätzlich kompilierte Versionen angeboten sowie eine umfassende Supportstrategie entwickelt werden. In diesem Zusammenhang sollte sich auch mit dem Thema Beiträge Dritter (siehe 3.2.3) beschäftigt werden.

- Wird eine Weiterentwicklung der Software durch die Community oder Partner angestrebt, ist dieses Thema zentral. Ein Prozess für Beiträge Dritter muss von Anfang an definiert sein und rechtliche Vereinbarungen (wie CLAs, siehe 3.2.3) vorbereitet werden. Die Relevanz der Supportstrategie sowie der Veröffentlichungsform hängt hier stark davon ab, wie entwicklungsaffin die Nutzenden sind. Umso weniger affin die Nutzenden sind, desto mehr muss ihnen mit vorkompilierten Dateien und umfangreichem Supportangebot entgegengegangen werden.

Ein weiterer wichtiger Punkt ist der Veröffentlichungsort. Dieser sollte entsprechend dem Standard in der Community gewählt werden. Klassische Optionen sind GitHub.com, Gitlab.com oder Zenodo.org. Eine Vorgabe vonseiten des DLR gibt es aktuell nicht. Jedoch muss sichergestellt werden, dass die Software nicht ausschließlich auf externen Systemen verfügbar ist.

[4] Software-Engineering-Empfehlungen des DLR, abrufbar unter: https://doi.org/10.5281/zenodo.1344607 (DE), https://doi.org/10.5281/zenodo.1344612 (EN)

print(“The free sharing and teaching of open source is incompatible with the notion of the solitary genius.”)

Golan Levin

3 Generelle Erläuterungen

Bevor auf die einzelnen Open-Source-Lizenzen im Detail eingegangen wird, soll an dieser Stelle zunächst ein allgemeiner Überblick zum Thema Open-Source-Software sowie den damit zusammenhängenden Themen verschafft werden.

3.1 Allgemeine Aspekte der Freien und Open-Source-Software

Zum besseren Verständnis des Lizenzmodells der Open-Source-Software im Allgemeinen sowie der einzelnen Lizenzpflichten im Besonderen werden in diesem Abschnitt einige allgemeine Aspekte erläutert. Dies umfasst die Definition und Entwicklungsgeschichte von Freier und Open-Source-Software ebenso wie Grundlagen von Urheber- und Patentrecht, Fragen des internationalen Lizenzrechts und die Besonderheiten des Copyleft.

3.1.1 Was ist Open-Source-Software?

Was Open-Source-Software ist, wird durch die Lizenz bestimmt. Es handelt sich um ein besonderes Lizenzmodell für Software, das sich besser verstehen lässt, wenn man die Entstehungsgeschichte kennt und das Lizenzmodell gegen andere Lizenzmodelle abgrenzt, insbesondere die „proprietäre" Lizenzierung.

3.1.1.1 Hintergrund der Entstehung des Lizenzmodells

Das Open-Source-Lizenzmodell, so wie wir es heute kennen, hat zwei parallele Entwicklungsstränge, die bemerkenswerterweise beide am Massachusetts Institute of Technology (MIT) Anfang der 1980er-Jahre ihren Ursprung nahmen. So hat eine Forschungsgruppe am MIT im Jahr 1983 eine Implementierung des Netzwerkprotokolls TCP/IP entwickelt und die beteiligten Forschenden hatten das primäre Ziel, diese Implementierung schnell und einfach in der Praxis einzuführen. Dieses Ziel überwog das Interesse, Lizenzgebühren zu generieren, sodass die Rechtsabteilung des MIT beauftragt wurde, eine einfache und schnelle Lizenzlösung zu finden. Ergebnis war die erste MIT License, die die Besonderheit für damalige Verhältnisse hatte, dass sie sich an jeden richtete, keine Unterschrift des Vertragspartners erforderlich war und zudem sehr weitreichende Nutzungsrechte einräumte. Man kann also von einem pragmatischen Ansatz sprechen, der heute noch für die sog. „Permissiven Lizenzen" typisch ist (siehe 3.1.3.2).

Nahezu gleichzeitig gründete der Programmierer Richard Stallman, der am MIT tätig war, das GNU-Projekt, das keinen pragmatischen, sondern einen politischen Ansatz hatte. Richard Stallman empfand es als eine Einschränkung seiner Freiheit als Programmierer, dass in den 1970er- und beginnenden 1980er-Jahren Softwarefirmen die Herausgabe des Quellcodes verweigerten und das Urheberrecht dazu einsetzten, Lizenzgebühren zu generieren. Software wurde damals zu einem eigenen Produkt, während sie zuvor ein wirtschaftlich nicht eigenständiger Bestandteil von Hardware-Produkten war und auf die eigenen Bedürfnisse angepasst werden konnte. GNU steht für „GNU is Not Unix" und hatte zum Ziel, technisch ein Unix-artiges Betriebssystem zu entwickeln, das aber nicht den lizenzrechtlichen Beschränkungen des klassischen Unix unterliegen sollte. Um zu verhindern, dass Dritte als bloße Profitierende auftreten und eigene Weiterentwicklungen eines GNU-Programms wieder unter herkömmlichen Lizenzbedingungen nutzen, erfand Richard Stallman das sog. „Copyleft". Beim Copyleft wird das Urheberrecht nicht dazu verwendet, Lizenzgebühren zu generieren, sondern dazu, die Entwicklungsfreiheit für alle abzusichern, indem Weiterentwicklungen unter der Ursprungslizenz freigegeben werden müssen (siehe 3.1.3). Erstmals umgesetzt wurde dieses Konzept in der GNU General Public License (GPL), die Ende der 1980er-Jahre Jahre veröffentlicht wurde.

Das GNU-Projekt und die zu seiner Unterstützung gegründete Organisation Free Software Foundation (FSF) nannten ihr Lizenzmodell „Free Software" (Freie Software) und bezogen sich damit auf die Nutzerfreiheiten. Lizenzen wie die MIT License wurden ebenfalls unter diesem Begriff gefasst, da sie, wenn auch ohne Copyleft, jeder Person umfassende Rechte zur Nutzung eines Programms einräumen. Erst Ende

der 1990er-Jahre, nachdem die Verbreitung von Freier Software durch die Entwicklung des Internets und zunehmende Modularisierung der Software-Programmierung eine erhebliche Ausbreitung erfahren hatte, wurde der Begriff „Open-Source-Software" geprägt. Dieser war zunächst vor allem ein Marketingbegriff, um das Konzept Freier Software auch in der Wirtschaft salonfähig zu machen, der sich aber schnell als generischer Begriff für dieses Lizenzmodell durchsetzte.

3.1.1.2 Definitionen für Freie Software und Open-Source-Software

Sowohl die Free Software Foundation mit der Free Software Definition[5] als auch die Open Source Initiative mit der Open Source Definition[6] legten früh Definitionen dafür vor, was als Freie bzw. Open-Source-Software verstanden wird. Die inhaltlichen Unterschiede dieser beiden Definitionen führen nicht zu abweichenden Ergebnissen bei der Bewertung. Sie machen beide deutlich, dass sich anhand der Lizenzierung entscheidet, ob etwas als Open-Source-Software anzusehen ist, und können synonym verwendet werden. Die wesentlichen Kriterien dafür sind:

- Die Lizenz muss die Bearbeitung der Software gestatten.
- Die Lizenz muss die Vervielfältigung und Verbreitung von bearbeiteten und unbearbeiteten Versionen der Software erlauben.
- Die Lizenz darf keine Lizenzgebühren für diese Nutzungshandlungen verlangen.
- Der Quellcode der Software muss öffentlich zugänglich sein.
- Es darf keine Beschränkung des Nutzendenkreises geben, das heißt, die Lizenzen müssen sich an jeden richten.
- Die Lizenz muss jegliche Nutzungszwecke erlauben, auch die kommerzielle Nutzung.

So darf Open-Source-Software verkauft werden und es dürfen bezahlte Dienstleistungen dafür angeboten werden. Nutzungshandlungen, wie die Vervielfältigung oder Verbreitung, dürfen jedoch nicht von Lizenzgebühren abhängig gemacht werden, weil dies die freie Nutzung beschränken würde.
Diese beiden Definitionen haben sich heute umfassend durchgesetzt. Insbesondere reicht es für die Qualifizierung als Open-Source-Software nicht aus, dass der Quellcode zugänglich ist, wenn nicht auch die entsprechenden Nutzungsrechte eingeräumt werden.

3.1.1.3 Abgrenzung zu anderen Lizenzmodellen

Zur Abgrenzung von dem Open-Source-Lizenzmodell hat sich der Begriff „proprietäre Software" eingebürgert. Hierunter werden herkömmliche Lizenzmodelle verstanden, z. B. die klassische Lizenzierung gegen Lizenzgebühren und auch das Freeware-Modell. Anders als bei „Free Software" werden bei Freeware keine umfassenden Nutzungsrechte eingeräumt, der Begriff „free" bezieht sich hier auf die Kostenfreiheit der angebotenen Programmkopie und nicht auf die Nutzerfreiheiten. Ein typisches Beispiel ist das Programm „Adobe Reader": Es darf zwar kostenlos heruntergeladen und auf einem eigenen Rechner eingesetzt werden, aber eine Weiterverbreitung oder Bearbeitung ist nicht gestattet.

Einen Sonderfall stellt „Public-Domain-Software" dar. Der Begriff stammt aus den USA und bezeichnet Software, an der kein Urheberrecht besteht. Dafür gibt es zwei typische Anwendungsfälle: So kann es von vornherein an einem urheberrechtlichen Schutz fehlen, wenn die Software von

[5] https://www.gnu.org/philosophy/free-sw.html.en
[6] https://opensource.org/docs/osd

US-Regierungsorganisationen stammt. Zudem kann ein Urheber seine Software als Public Domain freigeben, indem er auf sein Urheberrecht insgesamt verzichtet. Die Software wird damit gemeinfrei. Dies ist in anderen Rechtsordnungen, wie z. B. in den kontinentaleuropäischen Urheberrechtssystemen, nicht möglich, da das Urheberrecht dort auch eine persönlichkeitsrechtliche Komponente besitzt, die unverzichtbar ist (siehe 3.1.2.2). Dennoch kann eine explizite Public-Domain-Erklärung durchaus so verstanden werden, dass der Urheber eine unbeschränkte kostenlose Nutzung ermöglichen will. Dies würde dem Open-Source-Lizenzmodell weitgehend entsprechen.

3.1.2 Urheberrechtliche und patentrechtliche Grundlagen

Open-Source-Lizenzen dienen regelmäßig der vertraglichen Einräumung von urheberrechtlichen Nutzungsrechten. Daher werden zum besseren Verständnis Grundlagen des Urheberrechts dargestellt. In einigen wenigen Fällen können auch Patente mitlizenziert werden. Entsprechend findet sich nachfolgend auch ein Überblick zu dem Verhältnis von Patent- und Urheberrecht.

3.1.2.1 Schutzgegenstand

Der urheberrechtliche Schutz von Computerprogrammen ist in den §§ 69a ff. Urheberrechtsgesetz (UrhG) geregelt. Dabei wird nicht die Funktionalität eines Programms geschützt, sondern der konkrete Programmcode (sowohl Quellcode als auch Objektcode). Die Gerichte verlangen keine besonders ausgeprägte Schöpfungshöhe, sodass auch einfacher Programmcode, der nicht nur trivial ist, urheberrechtlichen Schutz genießt. Nicht als Computerprogramm ist hingegen die grafische Bedienoberfläche geschützt; diese kann allenfalls gesondert als Grafik urheberrechtlichen Schutz besitzen.

3.1.2.2 Wer ist Rechteinhaberin bzw. Rechteinhaber?

Inhaberin bzw. Inhaber des Urheberrechts ist die Person, die die Programmierung vorgenommen hat. Bei mehreren Beteiligten kann Miturheberschaft bestehen. Das Urheberrecht ist nicht übertragbar, da es eine persönlichkeitsrechtliche Komponente besitzt. Entsprechend kann ein Unternehmen nicht das Urheberrecht an sich erwerben, sondern gemäß § 69b UrhG nur die umfassenden (exklusiven) Verwertungsrechte an der Programmierung, die von seinen Arbeitnehmenden erstellt wurde. Dem Urheber verbleibt das Recht auf Namensnennung als persönlichkeitsrechtliche Position. Damit unterscheidet sich die Rechtslage in Deutschland von derjenigen in den USA. Dort erwirbt der Arbeitgeber unmittelbar das Copyright als sog. „work made for hire" und gilt als „Copyright owner".

Wird ein Programm iterativ entwickelt, das heißt, ein Entwickelnder modifiziert ein vorbestehendes Programm, dann erwirbt diejenige Person, die die Modifizierung vornimmt, ein sog. Bearbeiterurheberrecht. Das Bearbeiterurheberrecht ist ein von dem Urheberrecht an dem Originalprogramm abhängiges, aber selbstständiges Urheberrecht. Der Bearbeiterurheber benötigt die Gestattung des Originalurhebers zur Vornahme der Bearbeitung. Über die Verwertung des bearbeiteten Programms müssen sich dann Originalurheber und Bearbeiterurheber einigen. Dies geschieht bei der Open-Source-Lizenzierung regelmäßig dadurch, dass eine einheitliche Lizenz verwendet wird, die sowohl die Bearbeitung als auch die weitere Nutzung für jeden gestattet.

3.1.2.3 Nutzungsrechte und Lizenzen

Das Urheberrecht ist ein Verbotsrecht, daher kann die Inhaberin bzw. der Inhaber Dritten die Nutzung des Werks verbieten. Sofern anderen Personen die Nutzung gestattet wird, spricht man von einer Lizenz. Eine Lizenz ist letztlich nichts anderes als die vertragliche Einräumung von Nutzungsrechten. Nut-

zungsrechte können dabei einfach oder ausschließlich (exklusiv) gewährt werden. Die Inhaberin bzw. der Inhaber eines ausschließlichen Nutzungsrechts ist alleine befugt, ein Werk (z. B. ein Computerprogramm) zu nutzen. Bei der Einräumung von einfachen Nutzungsrechten können mehrere Berechtigte nebeneinander zur Nutzung befugt sein. Dies ist bei Open-Source-Lizenzen der Fall.

Des Weiteren kann bei einer Lizenz der räumliche, zeitliche und inhaltliche Nutzungsumfang beschränkt werden. Zudem können einzelne Nutzungsrechte, wie das Verbreitungsrecht, Vervielfältigungsrecht oder das Recht der öffentlichen Wiedergabe, gesondert und in einem unterschiedlichen Umfang eingeräumt werden. Bei Open-Source-Lizenzen wird ohne jegliche Beschränkung ein zeitlich, räumlich und inhaltlich unbeschränktes Recht eingeräumt, die Software in allen Nutzungsarten zu verwenden. Dies vereinfacht die Nachnutzung erheblich, weil stets sämtliche möglichen Nutzungshandlungen gestattet sind.

3.1.2.4 Erschöpfungsgrundsatz

Der Urheber besitzt das ausschließliche Verbreitungsrecht, das heißt das Recht, Kopien eines geschützten Programms an Dritte weiterzugeben. Hat der Urheber für eine konkrete Kopie von diesem Recht Gebrauch gemacht, „erschöpft" es sich. Dies bedeutet, dass der Vertrieb einer einmal rechtmäßig in Verkehr gebrachten Programmkopie von dem Urheber nicht weiter kontrolliert werden kann. Dieses Prinzip basiert auf dem Grundsatz der Freiheit des Warenverkehrs innerhalb des Europäischen Wirtschaftsraums (EWR) und sichert die Möglichkeit der Weiterveräußerung von Waren, insbesondere auch von gebrauchten Gütern (z. B. Büchern und Musik-DVDs). Dies gilt auch für Computerprogramme auf einem Datenträger und in einem Embedded System, selbst wenn die Programmkopie durch einen Download erworben wurde. Entsprechend darf auch Open-Source-Software, wenn eine konkrete Kopie rechtmäßig (also in Beachtung aller Lizenzbedingungen) erworben wurde, vom Empfangenden weiterveräußert werden, ohne dass dann die Lizenzbedingungen der entsprechenden Open-Source-Lizenz (nochmals) beachtet werden müssen. Grund dafür ist der Erschöpfungsgrundsatz, sodass für diese Handlung keine Lizenz benötigt wird. Allerdings wirkt sich der Erschöpfungsgrundsatz in der Praxis selten aus, da der Vertrieb einer rechtmäßig (lizenzkonform) in Verkehr gebrachten Kopie regelmäßig ebenfalls wieder die anwendbare Open-Source-Lizenz erfüllt.

3.1.2.5 Bestimmungsgemäße Benutzung

Auch wenn der Charakter des Urheberrechts als Verbotsrecht im Regelfall dazu führt, dass nur ein Lizenznehmer das Werk benutzen darf, gibt es auch gesetzlich geregelte Ausnahmen. Zu diesen urheberrechtlichen Schranken gehört die in § 69d UrhG geregelte Befugnis, eine rechtmäßig erworbene Programmkopie bestimmungsgemäß benutzen zu dürfen. Dies bedeutet, dass die Installation und das Laden in den Arbeitsspeicher gesetzlich gestattet sind und keiner besonderen Lizenz bedürfen (z. B. ein sog. „EULA"). Die Rechtslage ist in manchen anderen Ländern, wie den USA, abweichend. Die gesetzliche Ausnahme für die bestimmungsgemäße Benutzung rechtmäßig erworbener Kopien gilt auch für Open-Source-Software. Das heißt, dass für die reine Benutzung – also die interne Installation und den Ablauf eines Programms – keine Lizenzpflichten anfallen. Entsprechend liegt der Schwerpunkt der Analyse und Einhaltung von Open-Source-Lizenzen auf dem Vertrieb der Software an Dritte bzw. dem Angebot als Cloud-Service oder als Software as a Service (SaaS).

3.1.2.6 Verletzung von Open-Source-Lizenzen

Bei der Verletzung des Urheberrechts stehen dem Urheber zahlreiche Ansprüche zu. So kann er die Unterlassung der

verletzenden Handlung und Auskunft über Art und Umfang der Verletzung sowie Beseitigung der Verletzungsfolgen und Schadensersatz verlangen, wenn die Verletzung vorsätzlich oder fahrlässig vorgenommen wurde. Einige Open-Source-Lizenzen enthalten eine Klausel, die besagt, dass bei einer Lizenzverletzung die eingeräumten Nutzungsrechte automatisch wegfallen (z. B. Ziffer 4 GPL-2.0[7]). Die Nutzungsrechte werden damit auflösend bedingt eingeräumt, sodass aus einer Lizenzverletzung automatisch auch eine Urheberrechtsverletzung resultiert. Dies hat erhebliche praktische Konsequenzen: Der oder die verletzten Urheber können nicht nur verlangen, dass sich Nutzende, die die Lizenzbedingungen verletzen, an die Open-Source-Lizenz halten, sondern auch den weiteren Vertrieb der Software oder eines Produktes, das die Software beinhaltet, untersagen. Gerade bei hochpreisigen Produkten, wie z. B. Kraftfahrzeugen oder Maschinen, können so erhebliche finanzielle Einbußen bei scheinbar geringfügigen Lizenzverletzungen drohen, wie z. B. der fehlenden Mitlieferung von Lizenztexten. Der Einhaltung von Open-Source-Lizenzpflichten kommt damit besondere Bedeutung zu (zur Open-Source-Lizenz-Compliance siehe 3.1.6).

3.1.2.7 Verhältnis von Urheber- und Patentrecht

Computerprogramme sind das einzige Immaterialgut, das sowohl urheberrechtlich als auch patentrechtlich geschützt sein kann. Im Regelfall besteht nur urheberrechtlicher Schutz, der für die schöpferische Leistung der Programmiererin bzw. des Programmierers gewährt wird. Sofern ein Computerprogramm aber zusätzlich eine Erfindung implementiert, kann es auch patentrechtlichen Schutz genießen. Während der konkrete Programmcode durch das Urheberrecht geschützt wird, kann darüber hinaus eine Funktionalität patentrechtlich geschützt sein, wenn sie neu, erfinderisch und gewerblich anwendbar ist. Da der patentrechtliche Schutz auf der abstrakten Ebene der Funktionalität besteht, kann eine patentrechtlich geschützte Erfindung durch verschiedenen Programmcode implementiert werden. Dies bedeutet auch, dass eine Neuentwicklung versehentlich Drittpatente verletzen kann, weil diese dem Programmierenden nicht bekannt waren. Dies mag einer der Gründe sein, warum Softwarepatente bei Open-Source-Entwicklerinnen und -Entwicklern zumeist unbeliebt sind. Anders als Unternehmen sind sie zumeist nicht in der Lage, vor einer Entwicklung eine Patentrecherche durchzuführen.

Für die Nutzung von Open-Source-Software sind Patente in zweierlei Sicht relevant: Zum einen werden bei einer Open-Source-Lizenzierung eigene Patente, die durch die Open-Source-Software implementiert werden, in dem Umfang kostenlos mitlizenziert, der dem Nutzenden die Wahrnehmung der durch die Open-Source-Lizenz gewährten Rechte ermöglicht. Zum anderen besteht bei Open-Source-Software keine Gewähr der Rechteinhaberinnen und Rechteinhaber, dass die Software keine Patente Dritter verletzt. Der Nutzende muss daher selbst sicherstellen, dass die von ihr oder ihm vertriebene Software Patente Dritter beachtet. Bislang sind allerdings in der Praxis nur wenige Problemfälle bekannt geworden. Dies kann daran liegen, dass nicht jede innovative Software patentrechtlich geschützt werden kann, sondern nur dann, wenn ein über die Programmierung hinausreichender technischer Effekt erreicht wird. Dies kann insbesondere bei Betriebssystemsoftware und hardwarenahen Entwicklungen der Fall sein, weniger allerdings bei der reinen Anwendungsentwicklung. Ein typischer Gegenstand von Softwarepatenten sind Kompressionsalgorithmen, z. B. Codecs für Audio- oder Videokompressionsverfahren.

Open-Source-Lizenzen enthalten teilweise überhaupt keine speziellen Regelungen zur Patentlizenzierung. Dennoch wird der Lizenzgeber die erforderlichen Patente mitlizenzieren, da ansonsten ein Widerspruch zu den in der Lizenz gewährten Nutzungsrechten entstehen würde. Moderne Open-Source-Lizenzen enthalten zumeist Regelungen über Art und Umfang der Mitlizenzierung von Patenten (sofern existent), zum Teil

auch für den Fall, dass der Lizenznehmer Patente besitzt und die Software unter einer Open-Source-Lizenz an Dritte weitergibt. Sofern eigene Patente betroffen sein können, sollte daher stets eine Überprüfung der rechtlichen Auswirkungen einer Open-Source-Lizenz auf das eigene Patentportfolio stattfinden.

3.1.3 Inhalt und Auswirkungen eines Copyleft

Das Copyleft gehört zu den wichtigsten rechtlichen Aspekten bei Open-Source-Lizenzen. Es kann Auswirkungen auf die Lizenzierung von Eigenentwicklungen haben, aber auch für die Inkompatibilität von Open-Source-Lizenzen verantwortlich sein. Daher ist schon bei der Auswahl von Open-Source-Komponenten auf Copyleft-Effekte zu achten, da nicht alle Lizenzen ein solches Copyleft besitzen und die Reichweite des Copyleft unterschiedlich ausfallen kann.

3.1.3.1 Definition

Als Copyleft wird der rechtliche Effekt bezeichnet, dass Bearbeitungen eines vorbestehenden Programms ebenfalls wieder unter die Open-Source-Lizenz des Ursprungsprogramms gestellt werden müssen, wenn diese an Dritte weitergegeben werden. Das Copyleft ist eine Erfindung von Richard Stallman aus den 1980er-Jahren, um sicherzustellen, dass die Programme des GNU-Projektes stets Freie Software bleiben (siehe 3.1.1.1) und nicht durch Weiterentwicklungen Dritter *„reproprietarisiert"* werden können.[8] Dieses Lizenzkonzept wird dadurch umgesetzt, dass die entsprechenden Open-Source-Lizenzen eine Klausel enthalten, die verlangt, dass Bearbeitungen bei dem Vertrieb unter die Ausgangslizenz gestellt werden müssen. Ein Beispiel dafür ist Ziffer 2b der GPL-2.0:

"You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License."

3.1.3.2 Varianten des Copyleft

Historisch haben sich verschiedene Varianten des Copyleft entwickelt. Das sog. „starke Copyleft" verlangt, dass sämtliche Bearbeitungen und von dem Ursprungsprogramm abgeleiteten Werke der Ausgangslizenz unterstellt werden. Zu den „starken" Copyleft-Lizenzen gehören z. B. die GPL in allen Versionen (GPL-2.0, GPL-3.0), die GNU Affero General Public License, Version 3, (AGPL-3.0) und die Eclipse Public License, Version 1, (EPL-1.0).

Bei „schwachen" bzw. „beschränkten" Copyleft-Lizenzen müssen nicht sämtliche Bearbeitungen freigegeben werden, sondern nur solche, die in der jeweiligen Lizenz definiert sind. So beschränkt beispielsweise die Lesser General Public License (LGPL-2.1 und LGPL-3.0) das Copyleft auf die jeweilige unter der LGPL lizenzierte Bibliothek und verlangt nicht, dass damit verlinkte Programmkomponenten ebenfalls unter der LGPL lizenziert werden müssen. Die Mozilla Public License (MPL-2.0) besitzt ebenfalls ein „beschränktes" Copyleft, das auch als „file-based Copyleft" bezeichnet wird. Hier ist das Copyleft auf die jeweilige Quellcode-Datei beschränkt, die von dem Lizenznehmer verändert wird oder in die vorbestehender MPL-Code eingefügt wird.

[7] Als Abkürzungen für Open-Source-Lizenzen werden hier und im Folgenden die Identifier des SPDX-Standards verwendet, die Open-Source-Lizenzen eindeutig bezeichnen, vgl. https://spdx.org/licenses

[8] https://www.gnu.org/licenses/copyleft.html.en; https://copyleft.org

Es existieren auch Lizenzen ohne jegliches Copyleft, diese werden üblicherweise als „Permissive Lizenzen" bezeichnet (z. B. MIT, Apache-2.0 und BSD-3-Clause). Code unter solchen Lizenzen kann innerhalb von Eigenentwicklungen verwendet werden, ohne dass dieser als Open-Source-Software freigegeben oder im Quellcode zugänglich gemacht werden muss.

3.1.3.3 Bedeutung des Copyleft

Sofern Code unter Copyleft-Lizenzen in Neuentwicklungen verwendet wird, sind rechtlich zwei Aspekte besonders zu beachten. Zum einen ist zu prüfen, ob und in welchem Umfang selbst programmierter Code als Open-Source-Software unter der Ursprungslizenz freigegeben werden muss, zum anderen ist sicherzustellen, dass die verwendeten Open-Source-Komponenten keine inkompatiblen Lizenzen enthalten (siehe 3.1.4). Da das Copyleft in verschiedenen Varianten existiert (siehe 3.1.3.2), sind sowohl die Art des Copyleft als auch die konkrete technische Konstellation bei dieser Prüfung zu berücksichtigen. Für diese komplexe Aufgabe werden im Folgenden Hilfestellungen geleistet. Allerdings sollte bei der Verwendung von Software unter strengen Copyleft-Lizenzen stets eine Abstimmung unter Einbeziehung rechtlicher Beraterinnen und Berater erfolgen. Zudem ist zu beachten, dass die Urheber, die für ihre Entwicklungen eine Copyleft-Lizenz verwenden, im Regelfall ein besonderes Augenmerk darauf legen, ob die Nutzenden ihrer Software diesen Lizenzaspekt auch korrekt einhalten.

3.1.3.4 Technische Aspekte des strengen Copyleft

Strenge Copyleft-Lizenzen wie die GPL-2.0, die GPL-3.0, die AGPL-3.0 und die EPL-1.0 enthalten keine Definition dessen, was als „modification" oder „derivative work" zu verstehen ist und damit als Bearbeitung unter das Copyleft fällt. Die Lizenzen stellen damit auf den urheberrechtlichen Begriff der Bearbeitung ab, etwa § 3 UrhG bzw. die entsprechenden Regelungen in anderen Urheberrechtsordnungen. Das wirft die Folgefrage auf, was urheberrechtlich als Bearbeitung zu verstehen ist. Die Beurteilung wird noch dadurch verkompliziert, dass sich in den Urheberrechtsgesetzen keine näheren Definitionen dazu finden, was unter Bearbeitung im Softwareurheberrecht zu verstehen ist. Auch Rechtsprechung dazu existiert nahezu nicht. Insofern besteht in einigen Fragen noch eine Rechtsunsicherheit.

Dennoch existiert für einige typische technische Situationen ein weitgehendes Einverständnis in der Open-Source-Community, was unter das Copyleft fallen soll und was nicht. Dazu werden im Folgenden einige Faustregeln zusammengestellt:

- Änderungen im Quellcode einer Datei: Hier handelt es sich stets um eine Bearbeitung, die wieder der Ursprungslizenz unterliegt. Entsprechendes gilt für die Hinzufügung neuer Dateien, die zusammen mit bestehenden Dateien unter einer strengen Copyleft-Lizenz in eine Objektcode-Datei kompiliert werden.

- Verwendung von Entwicklungs-Tools (z. B. GNU C Compiler): Sofern solche Tools nicht selbst an Dritte weitergegeben werden, hat ihre Verwendung im Regelfall keine Auswirkung auf die damit entwickelte, kompilierte oder anderweitig erstellte Software. Zu beachten ist allerdings, dass manche Entwicklungstools eigenen Code in den sog. Target-Code einkopieren. Dann kann sich ein Copyleft-Effekt ergeben, wenn die Lizenz keine spezielle Ausnahme für diesen Fall vorsieht. Eine solche Ausnahme ist z. B. die Runtime Exception für den GNU C Compiler, die das Einkopieren einiger Bibliotheken während des Kompiliervorgangs in proprietären Code gestattet, ohne dass die Lizenzbedingungen dieser Bibliotheken beachtet werden müssen.[9]

- Inter-process-communication (IPC): Klassischerweise werden Kommunikationsschnittstellen als Abgrenzungskriterium zwischen selbstständigen Programmen verstanden. Somit führt die Verwendung von Software unter strengen Copyleft-Lizenzen regelmäßig dann nicht zu einem Copyleft-Effekt, wenn diese nur über IPC-Techniken wie z. B. Sockets oder Pipes angebunden wird (etwa für einen Datenaustausch über TCP/IP).

- Programmbibliotheken: Der wichtigste und umstrittenste Anwendungsfall ist die Verwendung von (A)GPL- bzw. EPL-1.0-Bibliotheken. Hier ist unklar, ob eine statische oder dynamische Verlinkung stets das Copyleft auslöst. Hierzu werden verschiedene Auffassungen vertreten, wobei für die (A)GPL überwiegend, insbesondere durch die FSF, angenommen wird, dass bei einer Verlinkung stets das Copyleft ausgelöst wird. Damit können keine GPL- oder AGPL-Bibliotheken verwendet werden, ohne das gesamte Programm unter die jeweilige Copyleft-Lizenz zu stellen. Bei der EPL-1.0, die überwiegend im Java-Umfeld Anwendung findet, wird vielfach eine andere Auffassung vertreten (siehe 4.7.1).

- Module und Plug-ins: Hier ist die Situation oftmals vergleichbar mit dynamisch verlinkten Bibliotheken, weil Komponenten mit spezieller Funktionalität zur Laufzeit in ein Programm eingebunden werden.

[9] Vgl. https://www.gnu.org/licenses/gcc-exception-3.1.html

3.1.4 Lizenzkompatibilität

Selbstständige Programme können jeweils ihre eigenen Open-Source-Lizenzen besitzen, denn deren Kompatibilität spielt keine Rolle, weil ein Copyleft des einen Programms keine Auswirkungen auf die Lizenz eines selbstständigen anderen Programms besitzt. Anders ist die Situation, wenn Programmcode unter verschiedenen Lizenzen in einem Programm integriert wird. Hier können Lizenzinkompatibilitäten entstehen, wenn Lizenzen verwendet werden, die sich gegenseitig ausschließende Lizenzbedingungen besitzen. Dies kann insbesondere bei Copyleft-Lizenzen der Fall sein (siehe 3.1.3.2): Wenn nämlich zwei Copyleft-Lizenzen verlangen, dass Bearbeitungen nur unter der Ursprungslizenz verbreitet werden dürfen, muss zwangsläufig eine der involvierten Copyleft-Lizenzen verletzt werden:

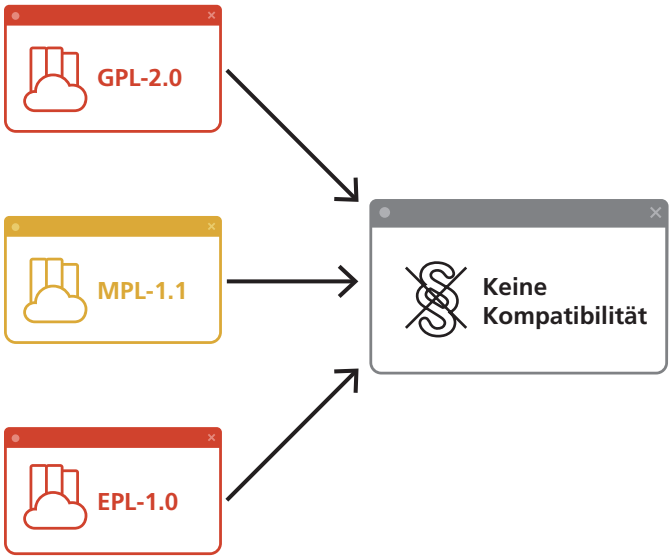


Abb. 1: Beispiel für eine Lizenzinkompatibilität, die durch die Kombination von Copyleft-Lizenzen mit sich gegenseitig ausschließenden Lizenzbedingungen entsteht

3.1.4.1 Kombination von Permissiven Lizenzen

Mangels Copyleft entfällt der Hauptgrund für Inkompatibilitäten. Bei den gängigen Permissiven Lizenzen sind keine Inkompatibilitäten aus anderen Gründen bekannt, sodass diese problemlos in einem Programm kombiniert werden können.

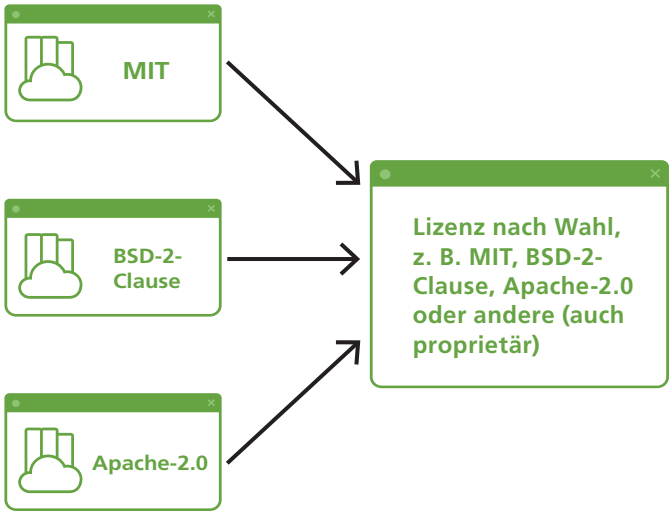


Abb. 2: Beispiel für die Kombination von permissiven Lizenzen. Hier liegt keine Lizenzinkompatibilität vor.

3.1.4.2 Kombination von Copyleft-Lizenzen

Copyleft-Lizenzen sind aus den oben genannten Gründen grundsätzlich inkompatibel zueinander. Eine Ausnahme gilt nur dann, wenn die involvierten Open-Source-Lizenzen eine Kompatibilitätsklausel beinhalten, die die konkrete Kombination gestattet. Die Kompatibilität besteht dann allerdings nur in eine Richtung, das heißt, die neue Software muss unter der Lizenz stehen, die die Kompatibilitätsklauseln zulassen. Dies ist jeweils im Einzelfall zu prüfen und wird nachfolgend an einem Beispiel dargestellt:

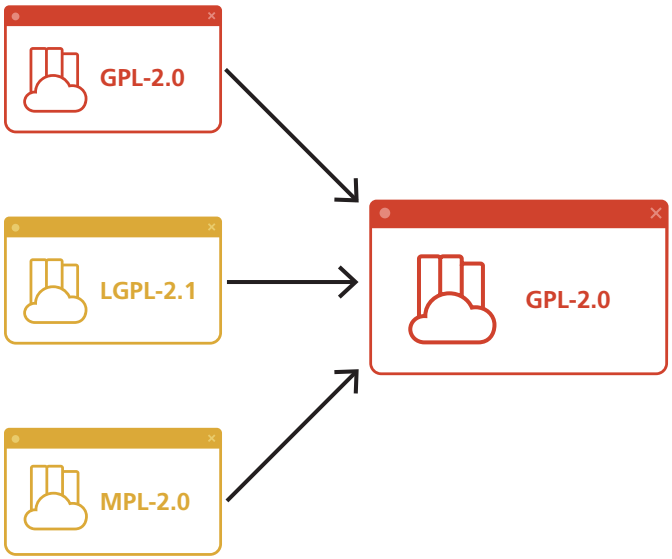


Abb. 3: Beispiel für die Kombination von Copyleft-Lizenzen. Aufgrund der Kompatibilitätsklauseln liegt hier keine Lizenzinkompatibilität vor.

3.1.4.3 Kombination einer Copyleft-Lizenz mit Permissiven Lizenzen

Sofern in einem Programm Code unter einer Copyleft-Lizenz und mehreren Permissiven Lizenzen verwendet wird, spricht zunächst nichts dagegen, das Ergebnis unter die Copyleft-Lizenz zu stellen. Allerdings ist es auch hier in Ausnahmefällen möglich, dass eine Permissive Lizenz Lizenzpflichten enthält, die die Copyleft-Lizenz nicht kennt und daher nicht zulässt. Denn „Copyleft" bedeutet auch, dass dem bearbeiteten Programm keine beliebigen Lizenzpflichten hinzugefügt werden dürfen, sondern dass nur die Lizenzpflichten der Copyleft-Lizenz zugelassen werden. Konsens besteht aber darüber, dass die Pflicht, verschiedene Lizenztexte mitliefern zu müssen, noch nicht zu einer Inkompatibilität führt.

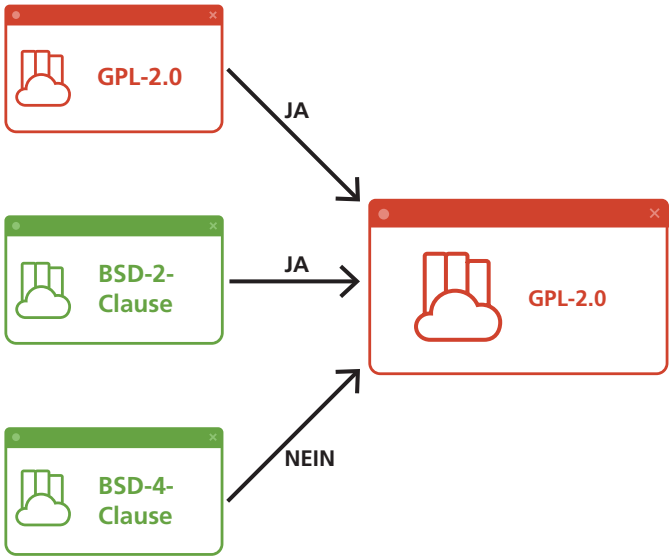


Abb. 4: Beispiel für eine Lizenzinkompatibilität, die durch die Kombination von Permissiven Lizenzen und Copyleft-Lizenzen entstehen kann

In dem vorstehenden Beispiel besteht keine Kompatibilität zwischen GPL-2.0 und BSD-4-Clause, weil diese Lizenz eine sog. Werbeklausel enthält, die die GPL-2.0 nicht kennt.[10] In den Fällen, in denen die Kompatibilität vom Einzelfall abhängt, sollte im Zweifelsfall Rücksprache mit den rechtlichen Beraterinnen und Beratern gehalten werden.

[10] https://www.gnu.org/licenses/bsd.html.en

Lizenzwahl bei dem Vertrieb veränderter Software

Der Prüfprozess für die Lizenzkompatibilität kann damit wie folgt beschrieben werden:

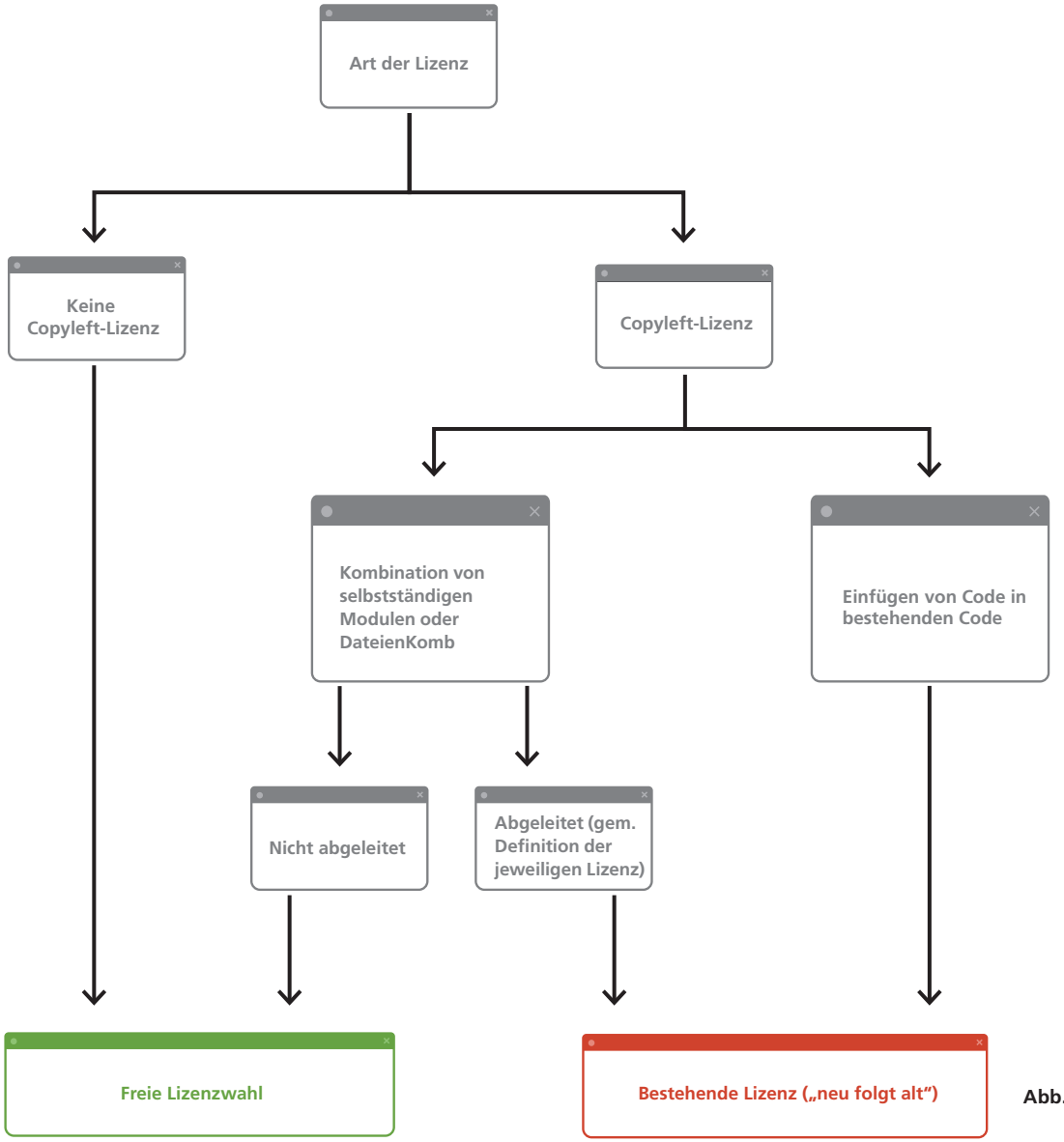


Abb. 5: Schematischer Prüfprozess zur Lizenzkompatibilität

Die nachfolgende Tabelle gibt einen Überblick über die Kompatibilität von wichtigen Open-Source-Lizenzen, unabhängig von der technischen Gestaltung. Es wird dabei nur die abstrakte Kompatibilität dargestellt, d. h. ob Code unter den betroffenen Lizenzen in allen technischen Varianten (also auch innerhalb einer Datei) kombiniert werden darf. Welche Lizenz dann als Ergebnislizenz verwendet werden muss, ergibt sich aus dem vorstehenden Diagramm.

	AGPL-3.0	GPL-2.0	GPL-3.0	EPL-1.0	EPL-2.0	LGPL-2.1	LGPL-3.0	CDDL-1.1	MPL-2.0	Apache-2.0	BSD-2/3-Clause	MIT	Zlib
AGPL-3.0		-	+	-	-	+	+	-	+	+	+	+	+
GPL-2.0	-		-	-	-	+	-	-	+	-	+	+	+
GPL-3.0	+	-		-	-	+	+	-	+	+	+	+	+
EPL-1.0	-	-	-		+	-	-	-	-	?[11]	+	+	+
EPL-2.0	-	-	-	+		-	-	-	-	?[11]	+	+	+
LGPL-2.1	+	+	+	-	-		+	-	+	-	+	+	+
LGPL-3.0	+	-	+	-	-	+		-	+	+	+	+	+
CDDL-1.1	-	-	-	-	-	-	-		-	-	+	+	+
MPL-2.0	+	+	+	-	-	+	+	-		?[11]	+	+	+
Apache-2.0	+	-	+	?[11]	?[11]	-	+	-	?[11]		+	+	+
BSD-2/3-Clause	+	+	+	+	+	+	+	+	+	+		+	+
MIT	+	+	+	+	+	+	+	+	+	+	+		+
Zlib	+	+	+	+	+	+	+	+	+	+	+	+	

Abb. 6: Übersicht der Lizenzkompatibilität ausgewählter Lizenzen

[11] Die Apache-2.0 verwendet eine abweichende Freistellungsklausel und verlangt in manchen Konstellationen die Mitlieferung einer NOTICE-Datei, das könnte als zusätzliche Lizenzpflicht angesehen werden.

3.1.5 Aspekte des internationalen Rechts

Open-Source-Software wird universell eingesetzt und Rechteinhaber bzw. Rechteinhaberinnen befinden sich in nahezu allen Staaten der Welt. Dies wirft Fragen des internationalen Rechts auf, für die nachfolgend ein kurzer Überblick gegeben wird. Dabei ist zu unterscheiden zwischen der Geltung der lokalen urheberrechtlichen Regelungen und der auf den Lizenzvertrag anwendbaren Rechtsordnung.

3.1.5.1 Internationales Urheberrecht

Es existiert kein einheitliches internationales Urheberrecht, sondern das Urheberrecht wird nach dem lokal anwendbaren Urheberrechtsgesetz für das jeweilige Territorium geregelt (Territorialitätsprinzip), ähnlich wie bei einem Patent, das auch nur von den staatlichen Patentämtern für das eigene Gebiet gewährt wird. Dies bedeutet, dass in Frankreich französisches Urheberrecht, in den USA US-amerikanisches Copyright und in Deutschland deutsches Urheberrecht gilt. Dennoch ist eine gewisse internationale Vereinheitlichung durch internationale Verträge und Abkommen erreicht worden. Die Revidierte Berner Übereinkunft (RBÜ) hat den Inländergrundsatz eingeführt, sodass auch ausländische Urheber im Inland wie inländische Urheber behandelt werden, wenn dies umgekehrt auch der Fall ist. Dies führt zu dem praktischen Ergebnis, dass Software-Programmierungen weltweit nach den jeweils lokalen Urheberrechtsordnungen geschützt sind. Der Urheber besitzt demnach nicht ein einheitliches international gültiges Urheberrecht, sondern ein Bündel nationaler Urheberrechte, die allerdings gewisse internationale Mindeststandards haben.

In der Europäischen Union ist durch die Computerprogramm-Richtlinie (2009/24/EG) das Softwareurheberrecht weitergehend harmonisiert.[12] Die Rechtslage ist insoweit in allen EU-Staaten einheitlich und wird durch die Rechtsprechung des Europäischen Gerichtshofs auch einheitlich interpretiert.

Unterschiede bestehen hingegen zwischen dem Copyright-System anglo-amerikanischer Staaten und dem Droit-d'Auteur-System kontinentaleuropäisch geprägter Staaten. Denn das Copyright-System ist nicht persönlichkeitsrechtlich geprägt (siehe 3.1.2.2), sodass auch Unternehmen originäre Copyright-Inhaber sein können. Das bekannte ©-Zeichen dient daher in den USA nicht der Benennung der Schöpferin bzw. des Schöpfers, sondern bezeichnet die Inhaberin bzw. den Inhaber der ausschließlichen Verwertungsrechte. Viele Open-Source-Lizenzen verlangen eine Weitergabe eines Copyright-Vermerks, der aber nicht mit einem Urhebervermerk verwechselt werden sollte, der die Autoren eines Programms bezeichnet. In vielen Fällen können Copyright-Vermerk und Urhebervermerk auch einheitlich sein, wenn Urheber und Inhaber der ausschließlichen Verwertungsrechte personenidentisch sind. Bei in Arbeitsverhältnissen entwickelter Software kann aber wie folgt unterschieden werden: Der Vermerk „© 2022 Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR), Autorin: Helga Musterfrau" benennt sowohl die Inhaberin/den Inhaber der ausschließlichen Verwertungsrechte als auch den die Urheberin/den Urheberder Programmierung. Für einen urheberrechtlichen Schutz ist allerdings weder ein Copyright-Vermerk noch ein Urhebervermerk erforderlich, er hat aber für den Nachweis der Rechtsinhaberschaft eine Bedeutung.

Nach dem anwendbaren Urheberrecht richtet sich, wer Inhaber oder Inhaberin der Rechte ist, der Umfang der urheberrechtlichen Positionen, die Dauer des Schutzes und die Rechtsfolgen bei Verletzungen. Gemäß Artikel 8 der Rom-II-Verordnung ist bei der Verletzung von Urheberrechten das Recht des Staates anzuwenden, für den der Schutz beansprucht wird. Entsprechend ist deutsches Urheberrecht anwendbar, wenn eine urheberrechtswidrige Vervielfältigung oder Verbreitung (auch) in Deutschland stattgefunden hat.

3.1.5.2 Internationales Vertragsrecht

Open-Source-Lizenzen haben aber nicht nur urheberrechtliche Relevanz, sondern sind auch Verträge zwischen Rechteinhaberinnen und Rechteinhabern und Nutzerinnen und Nutzern, auf die in den meisten Rechtsordnungen Vertragsrecht anwendbar ist. Wenn die Vertragspartner in verschiedenen Staaten ihren Sitz haben, stellt sich die Frage, welches nationale Vertragsrecht Anwendung findet. Da die meisten Open-Source-Lizenzen keine Rechtswahlklausel besitzen, kommt es gem. Art. 4 Abs. 2 Rom-I-Verordnung darauf an, in welchem Staat die Partei ihren Sitz hat, die die für den Vertrag „charakteristische" Leistung erbringt. Bei einem Lizenzvertrag ist das der Lizenzgeber. Demnach ist das Vertragsrecht am Sitz des Lizenzgebers anwendbar. Dies führt zu der Schwierigkeit, dass bei einer Vielzahl von Open-Source-Entwickelnden auch eine Vielzahl unterschiedlicher Rechtsordnungen Anwendung finden kann. Das Vertragsrecht ist insbesondere für die Regelungen zu Gewährleistung und Haftung, aber auch für die Art und Weise der Auslegung von Verträgen relevant.

3.1.6 Open Source License Compliance

Die Praxis hat gezeigt, dass die Einhaltung sämtlicher Open-Source-Lizenzen in einem komplexen Produkt regelmäßig nicht mehr ohne einen geeigneten Compliance-Prozess möglich ist. Elektronikkonzerne und Kraftfahrzeughersteller haben schon vor mehreren Jahren eigene Open-Source-Compliance-Abteilungen gegründet, um Open Source License Compliance sicherstellen zu können. Da bei der modernen Arbeitsteilung aber weiterhin auch Softwarekomponenten Dritter oder Zuliefererkomponenten in eigenen Produkten verwendet werden, hat sich gezeigt, dass gerade die Einhaltung von Lizenzpflichten bei zugelieferter Software oftmals mangelhaft ist. Zur Verbesserung der Situation wurde von einigen weltweit tätigen Unternehmen das OpenChain Project gegründet, das einen internationalen Standard für Open Source Compliance etabliert hat. Im Dezember 2020 wurde der OpenChain-Standard als ISO-Standard ISO/IEC 5230:2020 akzeptiert und veröffentlicht.[13]

Die Spezifikation des Standards sieht folgende wesentliche Elemente für den Compliance-Prozess vor:

- Eine schriftliche Open-Source-Policy mit einer Verteilung der Verantwortlichkeiten
- Das erforderliche Know-how der verantwortlichen Personen
- Kenntnis der einschlägigen Open-Source-Lizenzen in eigenen Softwareprodukten
- Kenntnis der Lizenzpflichten aus den einschlägigen Open-Source-Lizenzen
- Zugang zu rechtlicher Expertise
- Prozess zur Erstellung einer „Bill of Materials" der enthaltenen Open-Source-Software
- Prozess für die Einhaltung von Lizenzpflichten für typische Nutzungshandlungen
- Prozess für die Erstellung der Materialien (z. B. Lizenztexte, Urhebervermerke), die den Kundinnen und Kunden zu übergeben sind

[12] https://eur-lex.europa.eu/legal-content/DE/TXT/HTML/?uri=CELEX:32009L0024

[13] https://standards.iso.org/ittf/PubliclyAvailableStandards/c081039_ISO_IEC_5230_2020(E).zip

- Policy für Beiträge zu Open-Source-Projekten
- Bestätigung der Konformität des Standards

3.2 Veröffentlichung von Software als Open-Source-Software

Das Ziel von Open-Source-Software ist, dass Software durch Dritte genutzt, verändert und weiterverbreitet wird. Dazu gehört auch, neben der Schaffung der rechtlichen Rahmenbedingungen, die Software in eine entsprechende Form zu bringen und das dazugehörige Projekt bezüglich Themen wie Beiträge Dritter und Support zu strukturieren.

3.2.1 Aufbereitung für die Veröffentlichung

Software muss für die Veröffentlichung als Open-Source-Software aufbereitet werden. Das heißt, dass in jedem Fall der Lizenztext der gewählten Lizenz sowie ein Copyright-Hinweis hinzugefügt werden müssen. Je nach Veröffentlichungsform (siehe 3.2.2) und dem Einsatz des Codes Dritter müssen ggf. noch die entsprechenden Lizenzangaben mitgeliefert werden (siehe Kapitel 5).

Bezüglich der praktischen Umsetzung ist die Befolgung der REUSE-Spezifikation[14] zu empfehlen, welche zeigt, wie man menschen- und maschinenlesbar Urheber- und Lizenzinformationen an einer Software anbringt.

Neben der Pflicht zur Erfüllung rechtlicher Vorgaben empfiehlt es sich, die Software so aufzubereiten, dass Dritte diese einfach nutzen und modifizieren können.

Es empfiehlt sich, mindestens die folgenden Schritte zu befolgen:

- Nutzung eines Versionskontrollsystems

  Ein Versionskontrollsystem ist der zentrale Ort, an dem alle der Software zugehörigen Dateien abgelegt werden. Es ermöglicht das Nachvollziehen der Entwicklung sowie das Zurückkehren zu früheren Ständen der Software.

- Code in teilbarem Zustand

  Der Quellcode sollte für andere verständlich und nutzbar sein. Die Verwendung von Code-Kommentaren sowie die Strukturierung des Codes sind hier besonders hilfreich. Ideal ist es, übliche Standards der verwendeten Programmiersprache und/oder der Domäne zu befolgen.

- Essenzielle Dokumentation

  Es sollten zumindest aus Sicht der Nutzenden grundlegende Aspekte wie der grobe Funktionsumfang, Randbedingungen für den Einsatz der Software sowie Installations- und Nutzungsinformationen beschrieben werden. Je nach Art der Software und Zweck der Veröffentlichung kann die Erstellung von Dokumentation für weitere Zielgruppen (z. B. externe Beitragende, Mitentwickelnde) erforderlich sein.

- Stabile Version des Codes markieren

  Ein Release kommuniziert den Nutzenden, dass diese Version getestet und somit lauf- und funktionsfähig ist. Eine auf einem Releasenummer-Schema basierte Releasenummer vermittelt weitere Informationen über die Art des Releases. Ein sog. CHANGELOG enthält Informationen zu den Änderungen.

- Zitierfähigkeit (gilt für Forschungssoftware)

  Forschungssoftware hat einen akademischen Wert und sollte daher entsprechend behandelt werden. Dies beinhaltet die Möglichkeit zur Zitierung, die durch das Bereitstellen von Metadaten ermöglicht wird. Beispielsweise kann ein Forschungsdatenrepositorium zur langfristigen Archivierung, zur Speicherung der Zitationsmetadaten und zur Vergabe eines sog. Digital Object Identifier (DOI) genutzt werden.

Die vorstehende Liste basiert auf den Empfehlungen der Anwendungsklasse 1 der Software-Engineering-Empfehlungen des DLR (siehe 2.4) und wird in ähnlicher Form auch von anderen Institutionen empfohlen. Weitere Details zu den jeweiligen Schritten sowie ein praktisches Umsetzungsbeispiel sind in den Unterlagen der Schulung „Let's Make Your Script Ready for Publication“[15] enthalten.

3.2.2 Veröffentlichungsform

Die Wahl der Veröffentlichungsform hängt von der Zielgruppe und der gewünschten Verbreitung der Software ab.

Soll die Software für jeden einfach nutzbar sein, so sollte diese in vorkompilierter Form angeboten werden. Hierbei kann es, abhängig von der Art der Einbindung von (Open-Source)-Software Dritter, dazu kommen, dass diese dem Ergebnis hinzugefügt wird. Dadurch kommt es zum direkten Vertrieb fremden Codes. Somit müssen dessen Lizenzbedingungen analysiert und beachtet werden. Je nach Lizenz kann es dabei auch zu einer Beeinflussung der eigenen Lizenzwahl kommen.

Einfacher ist das Veröffentlichen der Software in Form des Quellcodes, jedoch erfordert dies, dass die Nutzenden qualifiziert sind, den Quellcode selbst zu kompilieren oder auszuführen. Bei dieser Veröffentlichungsform wird etwaige Dritt-Software in der Regel nicht mit übergeben, sondern lediglich eine Anleitung zu deren Bezug zur Verfügung gestellt.

Enthält die Anleitung konkrete Bezugsquellen, ist ebenfalls eine Lizenzanalyse nötig und die Lizenzen sollten bei der eigenen Lizenzwahl berücksichtigt werden. Es ist jedoch ausreichend, zu überprüfen, ob die angegebene Bezugsquelle alle Lizenzpflichten erfüllt (z. B. Auslieferung des Lizenztextes). Die Lizenzpflichten müssen jedoch nicht selbst erfüllt werden, da die Dritt-Software nicht selbst verbreitet wird oder zum Zeitpunkt der Verbreitung die Lizenzpflichten erfüllt werden. Von der Analyse regelmäßig ausgenommen werden können Systemvoraussetzungen, wie z. B. Interpreter sowie rein optionale Dritt-Software.

[14] https://reuse.software/spec

[15] Let's Make Your Script Ready for Publication:
https://gitlab.com/hifis/hifis-workshops/make-your-code-ready-for-publication/workshop-materials

Bei kleineren Skripten ist eine Veröffentlichung von Software als Quellcode üblich. Bei größeren Softwareprodukten ist oft eine Veröffentlichung in kompilierter Form empfehlenswert. Natürlich gibt es auch hier eine Reihe von Ausnahmen, insbesondere im Bereich von Software, die auf das zugrunde liegende System angepasst werden muss. Die Zielgruppe ist hier zumeist Fachpersonal und nicht der Consumer-Bereich.

Eine weitere Veröffentlichungs- und Vertriebsform sind Dockercontainer oder Dockerfiles. Bei Dockercontainern wird ein gesamtes System ausgeliefert, dementsprechend muss in Bezug auf die jeweiligen Lizenzpflichten auch das gesamte System berücksichtigt werden und nicht nur die in dem Dockercontainer laufende Anwendung.[16] Hierfür gibt es Werkzeuge, die bei der Analyse unterstützen.[17] Dockerfiles sind eine Art Rezept, nach welchem automatisiert von externen Quellen wie Docker Hub Code heruntergeladen und auf dem Zielrechner zusammengebaut wird. Auch der Vertrieb von Dockerfiles kann eine Verbreitung der referenzierten Software darstellen, sodass sicherzustellen ist, dass die Lizenzpflichten spätestens beim Zusammenbau erfüllt sind.[18] Entsprechend ist zu prüfen, ob die Quellen für die heruntergeladene Software die Lizenzpflichten für die herunterzuladende Software erfüllen. Bei der Auslieferung des Dockerfiles sollte ausdrücklich darauf verwiesen werden, wie die Lizenzinformationen erhalten werden können.

3.2.3 Beiträge Dritter

Die Einstellung zu Beiträgen Dritter hängt vom Kontext ab, in dem die Software erstellt wird. Während klassische Open-Source-Projekte in der Öffentlichkeit durch die Community entwickelt werden, gibt es auch viele, die in der Hand von Unternehmen oder Einzelnen liegen. In manchen Fällen findet die Entwicklung intern statt und nur der fertige Quellcode wird veröffentlicht. In diesem Fall sind Beiträge Dritter in der Regel nicht erwünscht. Generell können Beiträge Dritter wertvoll sein und das Projekt weiterbringen. Sie bedeuten gleichzeitig aber auch Aufwand und haben rechtliche Konsequenzen.

Erfolgt ein Beitrag zu einer Open-Source-Software, wird der beigetragene Code in der Regel der bestehenden Open-Source-Lizenz unterstellt und der Beitragende wird Bearbeiterurheber (siehe 3.1.2.2). Damit es durch die Beiträge zu keiner Lizenzverletzung kommt, muss sichergestellt werden, dass der beigetragene Code keine Rechte Dritter verletzt. Developer Certificates of Origin (DCO) oder Contributor License Agreements (CLAs) sind Optionen, sich dies vom Beitragenden versichern zu lassen. Mittels CLAs werden darüber hinaus einfache oder ausschließliche Verwertungsrechte vom Beitragenden an den Urheber der Software übertragen, um eine Relizenzierung unter abweichenden Lizenzbedingungen zu ermöglichen.
CLAs und DCO können jedoch eine Hürde für Beitragende darstellen und so mitunter abschreckend wirken. Daher verzichten etliche Open-Source-Projekte hierauf. Weiter bedeuten sie Aufwand bezüglich der Erstellung, Prüfung auf Vollständigkeit und Aufbewahrung. Übertragen jedoch Beitragende die Verwertungsrechte nicht an den Urheber der Software, muss z. B. im Falle einer Relizenzierung zuerst das Einverständnis aller Beitragenden eingeholt werden. Gibt es bei Lizenzen ohne Copyleft noch alternative Wege[19], so ist dies bei Copyleft-Lizenzen nicht der Fall. Eine Relizenzierung kann mit viel Aufwand verbunden sein, da ggf. Beitragende nicht mehr im Projekt aktiv und Kontaktdaten veraltet sind. Im Falle einer Ablehnung der Relizenzierung durch einen Beitragenden ist zu überprüfen, ob der jeweilige Beitrag aus der aktuellen Software entfernt werden kann.

Die Fragestellung der Wahrscheinlichkeit einer späteren Relizenzierung sowie die Chance des einfachen Ermöglichens von Beiträgen sind also ausschlaggebende Kriterien für die eigene Strategie im Umgang mit Beiträgen Dritter.

3.2.4 Support-Strategie

Support bedeutet Aufwand. Eine effiziente Support-Strategie liefert so viel Unterstützung wie nötig und ist auf die richtige Zielgruppe ausgerichtet (Nutzung, Entwicklung, Administration).

Grundlegende Empfehlungen sind eine ausreichende Dokumentation, für die Nutzergruppe relevante minimale Beispiele und die Möglichkeit zur Kontaktaufnahme mit den Entwickelnden. Letzteres kann z. B. mittels einer E-Mail-Adresse, einer Mailingliste oder eines sog. Issue Trackers umgesetzt werden.

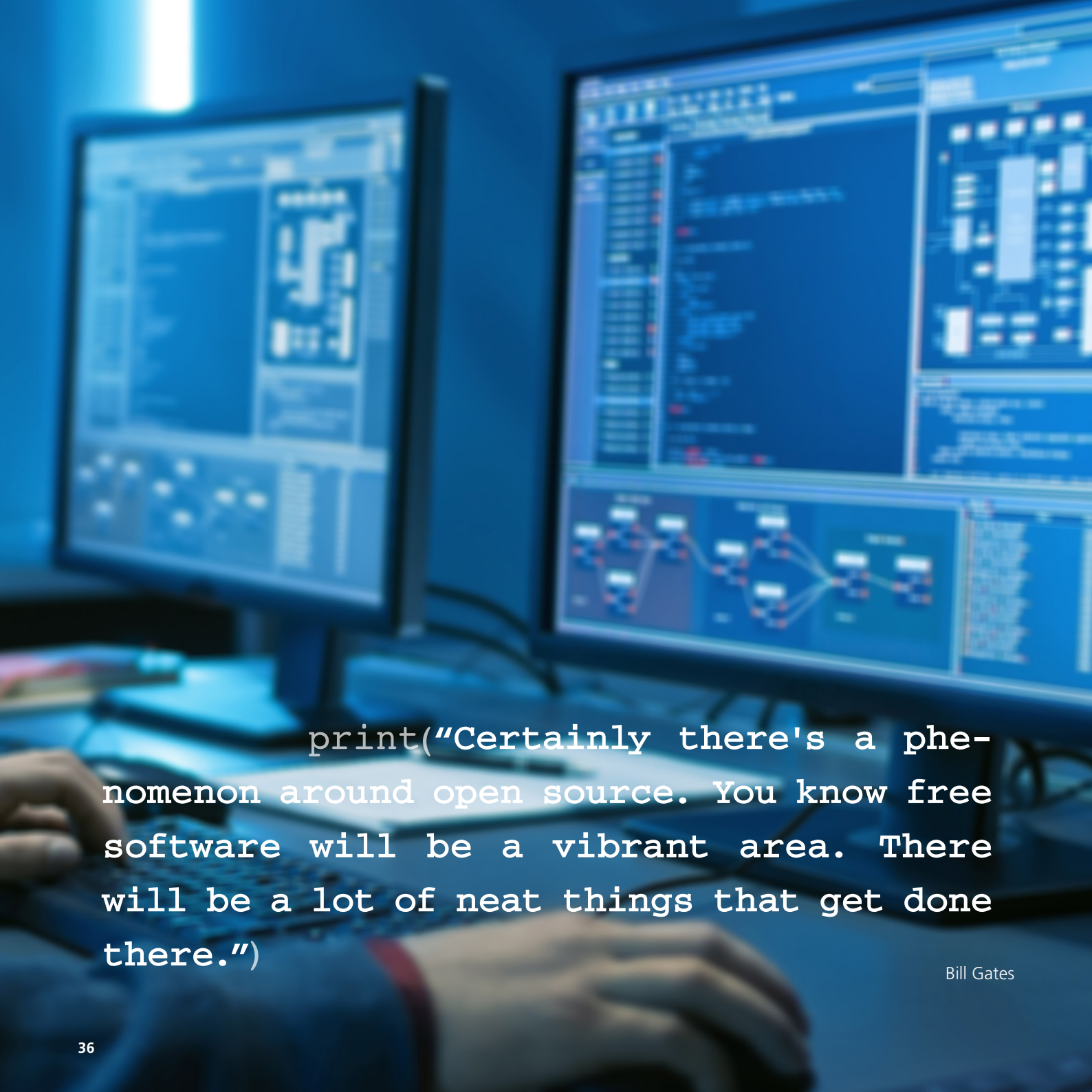
Viele Open-Source-Projekte setzen auf Community-Support, also gegenseitigen Support innerhalb der Nutzergruppen, und unterstützen dies z. B. durch den Betrieb von Foren oder nutzungsspezifischen Mailinglisten. Ist die Community für solche Aktivitäten zu klein, können umfangreiche Beispiel- und How-to-Sammlungen die Supportanfragen reduzieren. Größere Projekte bieten direkten Support oft nur kostenpflichtig an.

[16] Instruktiv zur License Compliance Hemel,
https://www.linuxfoundation.org/wp-content/uploads/Docker-Containers-for-Legal-Professionals-Whitepaper_042420.pdf

[17] Beispielsweise mit Hilfe des Open Source License Compliance Software Systems und Toolkit „FOSSOLOGY", https://www.fossology.org/ und dem Tool „Tern" zur Analyse, https://github.com/tern-tools/tern

[18] Vgl. dazu Jaeger, Distribution of Dockerfiles:
Who is responsible for FOSS Licence Compliance?, https://jolts.world/index.php/jolts/article/view/147

[19] Bei der Wahl einer neuen Lizenz werden die vorbestehenden Code-Bestandteile unter den bisherigen Lizenzen beibehalten, sodass ein Nebeneinander mehrerer Lizenzen existiert, deren Lizenzpflichten sämtlich erfüllt werden müssen.



```
print("Certainly there's a phenomenon around open source. You know free software will be a vibrant area. There will be a lot of neat things that get done there.")
```

Bill Gates

4 Open-Source-Lizenzen im Detail

Nachfolgend werden die wichtigsten Open-Source-Lizenzen und die jeweiligen Lizenzpflichten im Detail beschrieben. Bei der praktischen Umsetzung der Lizenzpflichten sind auch die Hinweise zu typischen Anwendungsfällen in Kapitel 5 sowie die Relevanz der Veröffentlichungsform (siehe 3.2.2) zu beachten.

4.1 GNU General Public License, Version 2 (GPL-2.0)

4.1.1 Überblick

Die GPL ist von überragender Bedeutung für die Welt Freier Software, da sie die erste Lizenz war, in der das Konzept des strengen Copyleft umgesetzt wurde (siehe 3.1.1.1). Die Version 1, der heute keine Bedeutung mehr zukommt, stammt aus dem Jahr 1989, die Version 2 wurde von der FSF im Jahr 1991 veröffentlicht und von Linus Torvalds für den Linux-Kernel übernommen. Standardmäßig ist in der GPL-2.0 vorgesehen, dass ihr unterstellte Computerprogramme auch unter später veröffentlichten Lizenzversionen (Ziffer 7, „or any later version") genutzt werden dürfen. Linus Torvalds hat allerdings deutlich gemacht, dass Linux nur unter der GPL-2.0 lizenziert ist[20], weshalb der Lizenz wegen der großen Verbreitung von Linux im Embedded-Bereich auch nach der Veröffentlichung der Version 3 noch große Bedeutung zukommt. Auch andere Open-Source-Projekte ziehen die GPL-2.0 der GPL-3.0 vor (z. B. MySQL), obwohl in ihr wegen des Alters einige Rechtsfragen nicht adressiert werden (z. B. Cloud Services, Digitale Signaturen).

4.1.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt nicht dazu, dass Lizenzpflichten erfüllt werden müssen. Ob die Bereitstellung von Funktionalität im Wege des SaaS einer Weiterverbreitung gleichzustellen ist, wird in der Lizenz nicht explizit geregelt und ist daher umstritten.[21]

- Mitlieferung des Lizenztextes
- Mitlieferung eines Disclaimers

  Hier empfiehlt es sich, den Mustertext aus dem Anhang der GPL-2.0 zu verwenden und gesondert in dem Dokument mit den Open-Source-Pflichtangaben aufzuführen („This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.")

- Mitlieferung von Copyright-Vermerken
- Zurverfügungstellung des Quellcodes

  Beim Vertrieb von GPL-2.0-Software im Objektcode muss der vollständige korrespondierende Quellcode entweder unmittelbar mitgeliefert werden oder es muss ein schriftliches Angebot beigefügt werden, den Quellcode jedem innerhalb von drei Jahren ab dem letzten Vertrieb des Objektcodes auf einem üblichen Datenträger zur Verfügung zu stellen, wobei Kostenerstattung für den Datenträger und dessen Versendung verlangt werden kann. Bei auf Linux basierenden Embedded-Systemen ist zu beachten, dass auch Informationen zur Toolchain zum „complete corresponding source code" gehören, damit der Nutzende einen Rebuild des Objektcodes vornehmen kann. Modifizierter Quellcode kann auch in Form von Patches zusammen mit dem Originalcode mitgeliefert werden. Ein Downloadangebot im Internet reicht nicht aus, es sei denn, der Objektcode wird ebenfalls über das Internet vertrieben.

- Änderungsvermerk

  Geänderte Quellcodedateien müssen mit einem Hinweis darauf versehen werden, dass diese modifiziert wurden und das Datum der Änderung. Wird der geänderte Code über mit dem Versionskontrollsystem Git an das Open-Source-Projekt übermittelt, bedarf es keines gesonderten Vermerkes, dass die entsprechenden Informationen in der History einsehbar sind. In allen anderen Fällen ist der Änderungsvermerk im Header aufzunehmen oder im Patch.

→ Hinweispflichten bei interaktiven Kommandos

Wenn eine GPL-2.0-Software Lizenzhinweise, Disclaimer oder Copyright-Vermerke anzeigt, dann müssen diese Hinweise auch in einem modifizierten Programm angezeigt werden, wenn dieses interaktive Kommandos einliest (z. B. über die Kommandozeile oder ein User Interface).

→ Copyleft

Bearbeitungen müssen ebenfalls der GPL-2.0 oder einer kompatiblen Lizenz unterstellt werden.[22] Zu diesem Zweck sind neu hinzugefügte Dateien mit einem Lizenzheader oder SPDX Identifier zu versehen.

4.2 GNU General Public License, version 3 (GPL-3.0)

4.2.1 Überblick

Die GPL-3.0 wurde von der Free Software Foundation im Jahr 2007 veröffentlicht, um auf neue technische und rechtliche Entwicklungen zu reagieren. Dazu gehört das rechtliche Verbot, technische Schutzmaßnahmen für urheberrechtlich geschützte Güter zu umgehen (z. B. Kopierschutztechnologien). Für GPL-3.0-lizenzierte Software verzichten Lizenzgeber und Lizenznehmer auf die Durchsetzbarkeit von solchen DRM-Systemen. Zudem muss bei Embedded-Systemen eine „Installation Information" mitgeliefert werden, die bei Verbraucherprodukten das Wiederaufspielen von modifizierten Versionen der Software ermöglicht (sog. „Anti-Tivoization-Klausel"). Die GPL-3.0 stellt klar, dass bei einem SaaS ohne Überlassung einer Programmkopie keine Lizenzpflichten zu erfüllen sind. Um den Lizenzgebern, die sich ein stärkeres Copyleft auch in der Cloud wünschen, entgegenzukommen, wurde die Schwesterlizenz GNU Affero General Public License (AGPL-3.0) veröffentlicht, die auch beim SaaS Lizenzpflichten vorsieht (siehe 4.3). Das Copyleft der GPL-3.0 entspricht weitgehend demjenigen der GPL-2.0.

Innovativ ist Ziffer 7 GPL-3.0, die spezielle Kompatibilitätsregelungen für andere Open-Source-Lizenzen vorsieht. Danach kann Code auch unter Permissiven Lizenzen, wie z. B. der Apache-2.0, mit GPL-3.0-Code kombiniert werden, sofern diese Lizenzpflichten enthalten, die in dem Katalog der Ziffer 7 explizit gestattet sind.[23]

4.2.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („conveying") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt ebenso wenig dazu, dass Lizenzpflichten erfüllt werden müssen, wie die Beauftragung von Drittfirmen, GPL-3.0 Programme für den internen Gebrauch anpassen zu lassen.

→ Mitlieferung des Lizenztextes

→ Beibehaltung von bereits bestehenden Disclaimern und Lizenzhinweisen

[20] Vgl. https://spdx.org/licenses/Linux-syscall-note.html

[21] Siehe Anwendungsfall 5.2.10

[22] Zur Reichweite des Copyleft und was als „Bearbeitung" („derivative work") anzusehen ist, siehe 3.1.3.4

[23] Siehe 3.1.4 zur Kompatibilität einzelner Lizenzen mit der GPL-3.0

- Mitlieferung von Copyright-Vermerken
- Zurverfügungstellung des Quellcodes

  Beim Vertrieb von GPL-3.0-Software im Objektcode muss der vollständige korrespondierende Quellcode dem Erwerbenden zur Verfügung gestellt werden. Dafür existieren verschiedene Optionen: Entweder kann der Quellcode unmittelbar mitgeliefert werden oder es wird ein Download-Link angeboten oder es muss ein schriftliches Angebot beigefügt werden, den Quellcode jedem innerhalb von drei Jahren ab dem letzten Vertrieb des Objektcodes und so lange, wie Ersatzteile oder Support für das Produkt angeboten werden, auf einem üblichen Datenträger zur Verfügung zu stellen, wobei Kostenerstattung für den Datenträger und dessen Versendung verlangt werden kann. Modifizierter Quellcode kann auch in Form von Patches zusammen mit dem Originalcode mitgeliefert werden. Zu beachten ist, dass bei Verbraucherprodukten auch eine „Installation Information" zum Quellcode gehört, die dem Erwerbenden die Neuinstallation der GPL-3.0-Software auf dem System ermöglicht (z. B. die Möglichkeit, eine eigene digitale Signatur zu verwenden, wenn das Produkt nur digital signierte Software installiert).
- Änderungsvermerk

  Geänderte Quellcodedateien müssen mit einem Hinweis darauf versehen werden, dass diese modifiziert wurden und mit dem Datum der Änderung. Wird der geänderte Code über das Versionskontrollsystem Git an das Open-Source-Projekt übermittelt, bedarf es keines gesonderten Vermerkes, dass die entsprechenden Informationen in der History einsehbar sind. In allen anderen Fällen ist der Änderungsvermerk im Header aufzunehmen oder im Patch.
- Hinweispflichten bei interaktiven Kommandos

  Wenn eine GPL-3.0-Software Lizenzhinweise, Disclaimer oder Copyright-Vermerke anzeigt, dann müssen diese Hinweise auch in einem modifizierten Programm angezeigt werden, wenn dieses interaktive Kommandos einliest (z. B. über die Kommandozeile oder ein User Interface).
- Copyleft

  Bearbeitungen müssen ebenfalls der GPL-3.0 oder einer kompatiblen Lizenz unterstellt werden.[24] Zu diesem Zweck sind neu hinzugefügte Dateien mit einem Lizenzheader oder SPDX Identifier zu versehen.

4.3 GNU Affero General Public License, Version 3 (AGPL-3.0)

4.3.1 Überblick

Die AGPL-3.0 wurde wie ihre „Schwesterlizenz" GPL-3.0 ebenfalls im Jahr 2007 veröffentlicht und unterscheidet sich von dieser nur in ihrer Ziffer 13, die die Nutzung im Wege des SaaS regelt. Demnach muss der Quellcode veränderter AGPL-3.0-Versionen zur Verfügung gestellt werden, wenn Kundinnen bzw. Kunden die Software interaktiv nutzen können („interacting with it remotely through a computer network"). Für Software, die nicht interaktiv genutzt werden kann (z. B. Webserver und Betriebssystem des Servers), bestehen keine Lizenzpflichten. Dies gilt auch für die Nutzung unveränderter AGPL-3.0-Software. Bei der Kombination mit GPL-3.0-Software ist zu beachten, dass die GPL-3.0- und AGPL-3.0-Bestandteile jeweils unter ihrer Ursprungslizenz bleiben, aber kombiniert genutzt und weiterverbreitet werden dürfen.

4.3.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („conveying") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt ebenso wenig dazu, dass Lizenz-

pflichten erfüllt werden müssen, wie die Beauftragung von Drittfirmen, AGPL-3.0-Programme für den internen Gebrauch anpassen zu lassen.

- Mitlieferung des Lizenztextes
- Beibehaltung von bereits bestehenden Disclaimern und Lizenzhinweisen
- Mitlieferung von Copyright-Vermerken
- Zurverfügungstellung des Quellcodes

  Beim Vertrieb und der SaaS-Nutzung von AGPL-3.0-Software im Objektcode muss der vollständige korrespondierende Quellcode dem Erwerbenden zur Verfügung gestellt werden. Dafür existieren verschiedene Optionen: Entweder kann der Quellcode unmittelbar mitgeliefert werden oder es wird ein Download-Link angeboten oder es muss ein schriftliches Angebot beigefügt werden, den Quellcode jedem innerhalb von drei Jahren ab dem letzten Vertrieb des Objektcodes und so lange, wie Ersatzteile oder Support für das Produkt angeboten werden, auf einem üblichen Datenträger zur Verfügung zu stellen, wobei Kostenerstattung für den Datenträger und dessen Versendung verlangt werden kann. Modifizierter Quellcode kann auch in Form von Patches zusammen mit dem Originalcode mitgeliefert werden. Zu beachten ist, dass bei Verbraucherprodukten auch eine „Installation Information" zum Quellcode gehört, die dem Erwerbenden die Neuinstallation der AGPL-3.0-Software auf dem System ermöglicht (z. B. die Möglichkeit, eine eigene digitale Signatur zu verwenden, wenn das Produkt nur digital signierte Software installiert). Bei einer SaaS-Nutzung bedarf es keiner „Installation Information".

- Änderungsvermerk

  Geänderte Quellcodedateien müssen mit einem Hinweis darauf versehen werden, dass diese modifiziert wurden und mit dem Datum der Änderung. Wird der geänderte Code über das Versionskontrollsystem Git an das Open-Source-Projekt übermittelt, bedarf es keines gesonderten Vermerkes, dass die entsprechenden Informationen in der History einsehbar sind. In allen anderen Fällen ist der Änderungsvermerk im Header aufzunehmen oder im Patch.

- Hinweispflichten bei interaktiven Kommandos

  Wenn eine AGPL-3.0-Software Lizenzhinweise, Disclaimer oder Copyright-Vermerke anzeigt, dann müssen diese Hinweise auch in einem modifizierten Programm angezeigt werden, wenn dieses interaktive Kommandos einliest (z. B. über die Kommandozeile oder ein User Interface).

- Copyleft

  Bearbeitungen müssen ebenfalls der AGPL-3.0 oder einer kompatiblen Lizenz unterstellt werden.[25] Zu diesem Zweck sind neu hinzugefügte Dateien mit einem Lizenzheader oder SPDX Identifier zu versehen.

[24] Siehe 3.1.4 zur Kompatibilität einzelner Lizenzen mit der GPL-3.0

[25] Zur Reichweite des Copyleft und was als „Bearbeitung" („Modification") anzusehen ist, siehe 3.1.3.4

4.4 GNU Lesser General Public License, version 2.1 (LGPL-2.1)

4.4.1 Überblick

Die erste Version der LGPL war die GNU Library General Public License, version 2.0 (LGPL-2.0), da sie 1991 zusammen mit der GPL-2.0 veröffentlicht wurde. Anlass für diese Lizenz war der Wunsch, für die GNU C Library (glibc) von der GPL-2.0 abzurücken, da von der FSF befürchtet wurde, dass die Anbietenden von proprietären Anwendungen für Linux eine eigene, proprietäre C Library entwickeln, die für freie Entwickelnde noch weniger Einfluss auf die Anwendungsebene von Linux („Userland") ermöglicht. Die Besonderheit aller LGPL-Versionen ist das schwache Copyleft, das auf die Bibliothek selbst beschränkt ist. Verlinkte Komponenten müssen, anders als bei der GPL, hingegen nicht unter der LGPL freigegeben werden. Dennoch gibt es auch für verlinkte Komponenten eine Reihe von Anforderungen, die in dieser Form nur die LGPL kennt (siehe unten die Lizenzpflichten der LGPL-2.1).

Die LGPL wurde 1999 in GNU Lesser General Public License, version 2.1, (LGPL-2.1) umbenannt, weil die FSF wegen der zunehmenden Beliebtheit der LGPL für Programmbibliotheken deutlich machen wollte, dass die GPL die vorzugswürdige Lizenz sei und die LGPL von geringerer Bedeutung („lesser") und nur in besonderen Konstellationen Anwendung finden solle. Des Weiteren wurde in der Version 2.1 auch die dynamische Verlinkung explizit geregelt, die in der Version 2.0 noch keinen Niederschlag gefunden hatte. Ansonsten sind die Versionen weitgehend identisch.

4.4.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt nicht dazu, dass Lizenzpflichten erfüllt werden müssen. Ob die Bereitstellung von Funktionalität im Wege des SaaS einer Weiterverbreitung gleichzustellen ist, wird in der Lizenz nicht explizit geregelt und ist daher umstritten. Die meisten Lizenzpflichten sind denjenigen der GPL-2.0 nachgestaltet. Eine Besonderheit der LGPL-2.1 besteht darin, dass dem Nutzenden der Austausch der Bibliothek ermöglicht werden muss, d. h. die Installation von modifizierten Versionen der Bibliothek.

- → Mitlieferung des Lizenztextes
- → Mitlieferung eines Disclaimers

  Hier empfiehlt es sich, den Mustertext aus dem Anhang der LGPL-2.1 zu verwenden und gesondert in dem Dokument mit den Open-Source-Pflichtangaben aufzuführen („This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
- → Mitlieferung von Copyright-Vermerken
- → Zurverfügungstellung des Quellcodes

  Beim Vertrieb von LGPL-2.1-Software im Objektcode muss der vollständige korrespondierende Quellcode entweder unmittelbar mitgeliefert werden oder es muss ein schriftliches Angebot beigefügt werden, den Quellcode jedem innerhalb von drei Jahren ab dem letzten Vertrieb des Objektcodes auf einem üblichen Datenträger zur Verfügung zu stellen, wobei Kostenerstattung für den Daten-

träger und dessen Versendung verlangt werden kann. Bei auf Linux basierenden Embedded-Systemen ist zu beachten, dass auch Informationen zur Toolchain zum „complete corresponding source code“ gehören, damit der Nutzende einen Rebuild des Objektcodes vornehmen kann. Modifizierter Quellcode kann auch in Form von Patches zusammen mit dem Originalcode mitgeliefert werden. Ein Downloadangebot im Internet reicht nicht aus, es sei denn, der Objektcode wird ebenfalls über das Internet vertrieben.

- Beibehaltung von bereits bestehenden Disclaimern und Lizenzhinweisen

Im Falle einer Modifikation der Bibliothek selbst müssen zusätzlich die folgenden Lizenzpflichten erfüllt werden:

- Die bearbeitete Software muss wieder eine Bibliothek sein.

  Damit soll sichergestellt werden, dass grundsätzlich auch eine Neuverlinkung möglich ist. Bei der Verwendung von LGPL-2.1-Code innerhalb einer Anwendung oder einer anderen Software, die keine Programmbibliothek ist, muss die bearbeitete Software gemäß Ziffer 3 unter der GPL-2.0 oder GPL-3.0 lizenziert werden.

- Änderungsvermerk

  Geänderte Quellcodedateien müssen mit einem Hinweis darauf versehen werden, dass diese modifiziert wurden und mit dem Datum der Änderung. Wird der geänderte Code über das Versionskontrollsystem Git an das Open-Source-Projekt übermittelt, bedarf es keines gesonderten Vermerkes, dass die entsprechenden Informationen in der History einsehbar sind. In allen anderen Fällen ist der Änderungsvermerk im Header aufzunehmen oder im Patch.

- Copyleft

  Bearbeitungen müssen ebenfalls der LGPL-2.1 oder einer kompatiblen Lizenz unterstellt werden.[26] Zu diesem Zweck sind neu hinzugefügte Dateien mit einem Lizenzheader oder SPDX Identifier zu versehen. Ziffer 3 enthält eine Kompatibilitätsklausel, die auch die Nutzung unter der GPL-2.0 oder GPL-3.0 gestattet, wenn die Lizenzheader entsprechend geändert und der Lizenztext der GPL-2.0 bzw. GPL-3.0 mitgeliefert wird.

Im Falle des gemeinsamen Vertriebs mit einer verlinkten Anwendung (oder einer anderweitig verlinkten Komponente) müssen gem. Ziffer 6 zusätzlich die folgenden Lizenzpflichten erfüllt werden:

- Gestattung der Bearbeitung der verlinkten Anwendung für interne Zwecke des Erwerbenden und des Reverse Engineerings zur Fehlerbehebung

  Zwar sieht die LGPL-2.1 kein Copyleft vor, stellt aber dennoch lizenzrechtliche Anforderungen an verlinkte Komponenten. Da die Bearbeitung eine urheberrechtlich relevante Handlung ist, muss die Gestattung explizit erfolgen, z. B. in der Lizenz der verlinkten Anwendung. Es muss allerdings nicht der Vertrieb der modifizierten Anwendung an Dritte gestattet werden.

- Hinweispflichten auf die Nutzung einer LGPL-Bibliothek

  In der Anwendung oder zusammen mit ihr muss darauf hingewiesen werden, dass eine LGPL-2.1-Bibliothek verwendet wird und der Lizenztext der LGPL-2.1 mitgeliefert wird. Wenn die Anwendung Copyright-Vermerke anzeigt, müssen auch die Copyright-Vermerke der LGPL-2.1-Bibliothek angezeigt werden sowie ein Hinweis, wo der Lizenztext der LGPL-2.1 zu finden ist.

[26] Zur Reichweite des Copyleft und was als „Bearbeitung“ („derivative work“) anzusehen ist, siehe 3.1.3.4

➜ Austausch der LGPL-2.1-Bibliothek ermöglichen

Dem Nutzenden soll es möglich sein, die LGPL-2.1-Bibliothek zu bearbeiten und die bearbeitete Bibliothek wieder mit der Anwendung neu zu verlinken. Dafür existieren verschiedene Optionen: Bei einer dynamischen Verlinkung reicht ein geeigneter „Shared-Library-Mechanismus" aus, sodass schnittstellenkompatible Bibliotheken einfach ausgetauscht werden können und dann weiterhin linken (z. B. in Linux-Systemen). Bei statischer Verlinkung ist dies nicht möglich, sodass eine Objektcode-Datei der Anwendung mitgeliefert werden muss oder ein schriftliches Angebot beizufügen ist, dass die Objektcode-Datei der Anwendung dem Nutzenden innerhalb von drei Jahren ab dem letzten Vertrieb zur Verfügung gestellt wird, wobei Kostenerstattung für den Datenträger und dessen Versendung verlangt werden kann.

Im Falle der Verwendung von Inhalten aus einem Headerfile der Bibliothek in der Anwendung, wenn diese Inhalte mehr als 10 Zeilen Code enthalten und nicht nur aus numerischen Parametern, Datenstrukturanordnungen und Zugriffsfunktionen bestehen:

➜ Lizenzierung unter der LGPL-2.1

Anders als die LGPL-3.0 verlangt die LGPL-2.1 dann eine Lizenzierung der Anwendung unter der LGPL, wenn funktionaler Code aus Headerfiles mit mehr als 10 Zeilen Code verwendet wird. Bei bis zu 10 Zeilen Code bestehen keine Lizenzpflichten.

4.5 GNU Lesser General Public License, version 3.0 (LGPL-3.0)

4.5.1 Überblick

Die Version 3 der LGPL wurde 2007 zusammen mit der GPL-3.0 veröffentlicht. Eine Besonderheit dieser Lizenz ist, dass sie als „Exception" zur GPL-3.0 ausgestaltet ist, d. h. auf den Lizenztext der GPL-3.0 wird insgesamt Bezug genommen und auf dieser Basis werden Ausnahmen formuliert, insbesondere zum Umfang des Copyleft. Wie schon bei den Versionen 2.0 und 2.1 müssen verlinkte Komponenten nicht unter der LGPL freigegeben werden, sondern können auch proprietär lizenziert werden.

4.5.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („conveying") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt ebenso wenig dazu, dass Lizenzpflichten erfüllt werden müssen, wie die Beauftragung von Drittfirmen, LGPL-3.0 Programme für den internen Gebrauch anpassen zu lassen.

➜ Mitlieferung der Lizenztexte

Da die LGPL-3.0 als „Exception" zur GPL-3.0 ausgestaltet ist und auf diese Bezug nimmt, müssen die Lizenztexte von GPL-3.0 und LGPL-3.0 mitgeliefert werden.

➜ Beibehaltung von bereits bestehenden Disclaimern und Lizenzhinweisen

➜ Mitlieferung von Copyright-Vermerken

➜ Zurverfügungstellung des Quellcodes

Beim Vertrieb von LGPL-3.0-Software im Objektcode muss der vollständige korrespondierende Quellcode der Bibliothek dem Erwerbenden zur Verfügung gestellt wer-

den. Dafür existieren verschiedene Optionen: Entweder kann der Quellcode unmittelbar mitgeliefert werden oder es wird ein Download-Link angeboten oder es muss ein schriftliches Angebot beigefügt werden, den Quellcode jedermann innerhalb von drei Jahren ab dem letzten Vertrieb des Objektcodes und so lange, wie Ersatzteile oder Support für das Produkt angeboten werden, auf einem üblichen Datenträger zur Verfügung zu stellen, wobei Kostenerstattung für den Datenträger und dessen Versendung verlangt werden kann. Modifizierter Quellcode kann auch in Form von Patches zusammen mit dem Originalcode mitgeliefert werden. Zu beachten ist, dass bei Verbraucherprodukten auch eine „Installation Information" zum Quellcode gehört, die dem Erwerbenden die Neuinstallation der LGPL-3.0-Bibliothek auf dem System ermöglicht (z. B. die Möglichkeit, eine eigene digitale Signatur zu verwenden, wenn das Produkt nur digital signierte Software installiert).

Im Falle einer Modifikation der Bibliothek selbst müssen zusätzlich die folgenden Lizenzpflichten erfüllt werden:

- Änderungsvermerk

  Geänderte Quellcodedateien müssen mit einem Hinweis darauf versehen werden, dass diese modifiziert wurden und mit dem Datum der Änderung. Wird der geänderte Code über das Versionskontrollsystem Git an das Open-Source-Projekt übermittelt, bedarf es keines gesonderten Vermerkes, dass die entsprechenden Informationen in der History einsehbar sind. In allen anderen Fällen ist der Änderungsvermerk im Header aufzunehmen oder im Patch.

- Copyleft

  Bearbeitungen müssen ebenfalls der LGPL-3.0 oder einer kompatiblen Lizenz unterstellt werden.[27] Zu diesem Zweck sind neu hinzugefügte Dateien mit einem Lizenzheader oder SPDX Identifier zu versehen. Ziffer 7 GPL-3.0 enthält eine Kompatibilitätsklausel, die die Streichung von „Additional Permissions" wie in der LGPL-3.0 und damit auch die Nutzung unter der GPL-3.0 gestattet (z. B. zusammen mit einer GPL-3.0-Anwendung). Dazu sollten die Lizenzheader entsprechend geändert und der Lizenztext der LGPL-3.0 nicht mehr mitgeliefert werden.

Im Falle des gemeinsamen Vertriebs mit einer verlinkten Anwendung (oder einer anderweitig verlinkten Komponente) müssen gem. Ziffer 4 zusätzlich die folgenden Lizenzpflichten erfüllt werden:

- Keine Beschränkung der Bearbeitung der Bibliothek als Teil der Kombination mit der verlinkten Anwendung und des Reverse Engineerings zur Fehlerbehebung

  Anders als bei der LGPL-2.1 muss keine Bearbeitung der verlinkten Komponenten gestattet werden, allerdings darf die Bearbeitung der Bibliothek (als Teil der Kombination von Anwendung und Bibliothek) auch nicht beschränkt werden. Zudem muss das Reverse Engineering zur Fehlerbehebung solcher Bearbeitungen zulässig sein. Auch wenn § 69d UrhG Fehlerbehebungen gesetzlich erlaubt, sollte im Hinblick auf abweichende Urheberrechtsregelungen im Ausland das Reengineering explizit gestattet werden.

- Hinweispflichten auf die Nutzung einer LGPL-Bibliothek

  In der Anwendung oder zusammen mit ihr muss darauf hingewiesen werden, dass eine LGPL-3.0-Bibliothek verwendet wird und die Lizenztexte der LGPL-3.0 und GPL-3.0 mitgeliefert werden. Wenn die Anwendung Copyright-Vermerke anzeigt, müssen auch die Copyright-Vermerke der LGPL-3.0-Bibliothek angezeigt werden sowie ein Hinweis, wo die Lizenztexte der LGPL-3.0 und GPL-3.0 zu finden sind.

- Hinweispflichten bei interaktiven Kommandos

  Wenn eine verlinkte Anwendung Lizenzhinweise, Disclaimer oder Copyright-Vermerke anzeigt, dann müssen diese Hinweise auch in einem modifizierten Programm angezeigt werden, wenn dieses interaktive Kommandos einliest (z. B. über die Kommandozeile oder ein User Interface).

[27] Zur Reichweite des Copyleft und was als „Bearbeitung" („derivative work") anzusehen ist, siehe 3.1.3.4

→ Austausch der LGPL-3.0-Bibliothek ermöglichen

Dem Nutzenden soll es möglich sein, die LGPL-3.0-Bibliothek zu bearbeiten und die bearbeitete Bibliothek wieder mit der Anwendung neu zu verlinken. Dafür existieren verschiedene Optionen: Bei einer dynamischen Verlinkung reicht ein geeigneter „Shared-Library-Mechanismus" aus, sodass schnittstellenkompatible Bibliotheken einfach ausgetauscht werden können und dann weiterhin linken (z. B. in Linux-Systemen). Bei statischer Verlinkung ist dies nicht möglich, sodass eine Objektcode-Datei der Anwendung mitgeliefert werden muss oder ein schriftliches Angebot beizufügen ist, dass die Objektcode-Datei der Anwendung dem Nutzenden innerhalb von drei Jahren ab dem letzten Vertrieb des Objektcodes und so lange, wie Ersatzteile oder Support für das Produkt angeboten werden, auf einem üblichen Datenträger zur Verfügung gestellt wird, wobei Kostenerstattung für den Datenträger und dessen Versendung verlangt werden kann. Zudem soll dem Nutzenden die Neuverlinkung explizit gestattet werden, was z. B. in den Lizenzbedingungen der verlinkten Anwendung geschehen kann.

Im Falle der Verwendung von Inhalten aus einem Headerfile der Bibliothek in der Anwendung, wenn diese Inhalte mehr als 10 Zeilen Code enthalten und nicht nur aus numerischen Parametern, Datenstrukturanordnungen und Zugriffsfunktionen bestehen:

→ Hinweispflicht auf die LGPL-3.0-Bibliothek

Auch wenn die Bibliothek selbst nicht zusammen mit der Anwendung vertrieben wird, muss ein auffälliger Hinweis erfolgen, dass die LGPL-3.0-Bibliothek in der Anwendung verwendet wird und dass diese unter der LGPL-3.0 lizenziert ist. Zudem müssen die Lizenztexte von LGPL-3.0 und GPL-3.0 mitgeliefert werden.

4.6 Mozilla Public License, version 2 (MPL-2.0)

4.6.1 Überblick

Die MPL-2.0 stammt aus dem Jahr 2012 und ist die aktualisierte Version der Vorgängerlizenzen MPL-1.0 und MPL-1.1. Sie besitzt ein schwaches Copyleft, das auch als „file-based" Copyleft bezeichnet wird und sich auf Quellcode-Dateien beschränkt, die MPL-Code enthalten. Die MPL wurde ursprünglich für die Programme Firefox und Thunderbird entworfen, die von der Mozilla Foundation auf der Basis des Netscape Navigator entwickelt wurden. Sie wird aber auch für zahlreiche andere Open-Source-Programme verwendet und ist zudem das Vorbild für weitere Lizenzen wie die Common Distribution und Development License (CDDL).

Die MPL-2.0 enthält eine Kompatibilitätsklausel zugunsten von GPL-2.0, GPL-3.0, LGPL-2.1, LGPL-3.0 und AGPL-3.0. Danach darf MPL-2.0-Code, der mit Programmcode einer der GNU-Lizenzen kombiniert wird, unter den Bedingungen der entsprechenden GNU-Lizenz weiterverbreitet werden.[28]

4.6.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt nicht dazu, dass Lizenzpflichten erfüllt werden müssen.

→ Mitlieferung des Lizenztextes oder einer Referenz auf den Lizenztext

→ Beibehaltung von bereits bestehenden Disclaimern und Lizenzhinweisen

→ Zurverfügungstellung des Quellcodes

Beim Vertrieb von MPL-2.0-Software im Objektcode muss der korrespondierende Quellcode dem Erwerbenden zugänglich gemacht werden. Die Lizenz lässt offen, wie dies zu geschehen hat und macht nur die Vorgabe, dass dies zeitnah, auf angemessene Weise und zu Kosten, die die Kosten für den Vertrieb nicht übersteigen, geschehen muss. Ein Download-Link reicht damit aus.

→ Copyleft

Bearbeitungen von vorbestehenden MPL-2.0-Dateien oder Dateien, in die MPL-2.0-Code einkopiert wird, müssen ebenfalls der MPL-2.0 oder einer kompatiblen Lizenz unterstellt werden.[29]

Die EPL-1.0 erlaubt auch die Nutzung unter einer neueren Lizenzversion wie der EPL-2.0, die Eclipse Foundation gibt Hinweise zur Umsetzung eines Lizenzwechsels.[30] Die EPL-2.0 ist standardmäßig nicht kompatibel mit den GNU-Lizenzen, es sei denn, der Quellcode enthält ein entsprechendes Exhibit A wie am Ende des Lizenztextes der EPL-2.0 beschrieben.[31]

4.7.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt nicht dazu, dass Lizenzpflichten erfüllt werden müssen.

4.7 Eclipse Public License, version 2 (EPL-2.0)

4.7.1 Überblick

Der EPL-2.0 kommt insbesondere im Java-Umfeld große Bedeutung zu. Anders als noch die EPL-1.0, die ein strenges Copyleft besaß, ist die EPL-2.0 eine schwache Copyleft-Lizenz, die das Copyleft auf Programm oder Bibliotheken beschränkt, aber sich nicht auf verlinkte Komponenten erstreckt. Das Copyleft ähnelt damit der LGPL, macht aber keine Vorgaben für verlinkte Komponenten.

→ Mitlieferung des Lizenztextes

→ Beibehaltung von bereits bestehenden Disclaimern und Lizenzhinweisen

→ Zurverfügungstellung des Quellcodes

Beim Vertrieb von EPL-2.0-Software im Objektcode muss der korrespondierende Quellcode dem Erwerbenden zugänglich gemacht werden. Entweder kann der Quellcode direkt mitgeliefert werden oder der Erwerbende muss darüber informiert werden, wie er den Quellcode

[28] Zur Umsetzung siehe https://www.mozilla.org/en-US/MPL/2.0/combining-mpl-and-gpl

[29] Zur Reichweite des Copyleft und was als „Bearbeitung" („Modification") anzusehen ist, siehe 3.1.3.4

[30] Vgl. https://www.eclipse.org/legal/epl-2.0/faq.php#h.tci84nlsqpgw

[31] Dort heißt es: „Exhibit A - Form of Secondary Licenses Notice - "This Source Code may also be made available under the following Secondary Licenses when the conditions for such availability set forth in the Eclipse Public License, v. 2.0 are satisfied: {name license(s), version(s), and exceptions or additional permissions here}." - Simply including a copy of this Agreement, including this Exhibit A is not sufficient to license the Source Code under Secondary Licenses."

auf angemessene Weise durch einen üblichen Weg der Softwareüberlassung (z. B. Datenträger oder Download) erhalten kann.

➔ Copyleft

Bearbeitungen von EPL-2.0-Programmen oder -Bibliotheken müssen ebenfalls der EPL-2.0 unterstellt werden. Verlinkte Komponenten, die selbst geschrieben wurden, können abweichende Lizenzbedingungen haben. Das ergibt sich aus den Definitionen von „Contribution" und „Modified Works" in Ziffer 1 EPL-2.0.

4.8 BSD-2-Clause License und BSD-3-Clause License (BSD-2-Clause und BSD-3-Clause)

4.8.1 Überblick

Die originale BSD-Lizenz (BSD-4-Clause) wurde Ende der 1980er-Jahre von der University of California für die „Berkeley Software Distribution" entwickelt und ist eine Permissive Lizenz ohne Copyleft. Entsprechend kann Code unter den BSD-Lizenzen zusammen mit Eigenentwicklungen verwendet werden, ohne dass eine Pflicht zur Freigabe als Open-Source-Software besteht. Sie wird als „BSD-4-Clause" bezeichnet, weil sie vier Lizenzklauseln besitzt. Um Kompatibilität mit der GPL-2.0 herzustellen, wurde die Klausel Nr. 3 (Werbeklausel) aus der originalen Lizenz gestrichen, sodass die BSD-3-Clause entstanden ist. Da die BSD-Lizenzen sog. Template-Lizenzen sind, die von jedem für eigene Programme verwendet werden können, wenn ein entsprechend geänderter Copyright-Vermerk verwendet wird, existiert auch noch die BSD-4-Clause. Zur weiteren Vereinfachung wurde von manchen Open-Source-Projekten auch ein ursprünglich in Klausel Nr. 4 enthaltenes Werbeverbot mit dem Namen des Lizenzgebers gestrichen, dies wird dann als BSD-2-Clause bezeichnet, weil nur noch zwei der ursprünglichen vier Lizenzklauseln übrig geblieben sind.

4.8.2 Lizenzpflichten

Sämtliche Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt zu keinen Lizenzpflichten.

➔ Mitlieferung des Lizenztextes

Es muss der Lizenztext mitsamt Disclaimer und vorangestellter Copyright-Notice mitgeliefert werden. Da viele Programme nicht nur einen Copyright-Vermerk enthalten und auch innerhalb des Lizenztextes bzw. Disclaimers auf die Rechteinhaberin bzw. den Rechteinhaber referenziert wird, reicht es meist nicht aus, eine BSD-Lizenz mitzuliefern, sondern es müssen alle vorhandenen BSD-Lizenzen extrahiert und mitgeliefert werden.

➔ Werbeverbot

Das Verbot, bearbeitete Programme mit dem Namen der Rechteinhaberinnen bzw. Rechteinhaber zu bewerben, gilt explizit nur bei der BSD-3-Clause, ist aber aus allgemeinen wettbewerbsrechtlichen Gründen auch für die BSD-2-Clause und andere Lizenzen zu beachten.

4.9 MIT License (MIT)

4.9.1 Überblick

Die MIT License wurde von dem Massachusetts Institute of Technology Mitte der 1980er-Jahre in einer Vorläuferform entwickelt, um Netzwerkprotokolle einfach und schnell im Markt

durchzusetzen, ohne schriftliche Lizenzverträge aufsetzen zu müssen. Als einfache und unkomplizierte Permissive Lizenz ohne Copyleft ist sie inzwischen zur verbreitetsten Open-Source-Lizenz geworden, die mit anderen Open-Source-Lizenzen kompatibel ist und auch unproblematisch in proprietären Eigenentwicklungen verwendet werden kann.

4.9.2 Lizenzpflichten

Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt zu keinen Lizenzpflichten.

- Mitlieferung des Lizenztextes

  Es muss der Lizenztext mitsamt vorangestellter Copyright-Notice mitgeliefert werden. Da viele Programme nicht nur einen Copyright-Vermerk enthalten, reicht es meist nicht aus, eine MIT-Lizenz mitzuliefern, sondern es müssen alle vorhandenen MIT-Lizenzen extrahiert und mitgeliefert werden.

4.10 zlib License (Zlib)

4.10.1 Überblick

Die zlib License wird seit 1995 für die Kompressionssoftware gleichen Namens verwendet. Es handelt sich um eine einfache und unkomplizierte Permissive Lizenz ohne Copyleft, die mit anderen Open-Source-Lizenzen kompatibel ist und auch unproblematisch in proprietären Eigenentwicklungen verwendet werden kann. Sie sollte nicht mit der „zlib/libpng License with Acknowledgement" verwechselt werden. Beim Vertrieb im Objektcode sind keine Lizenzpflichten zu erfüllen.

4.10.2 Lizenzpflichten

Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution") von Quellcode-Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt zu keinen Lizenzpflichten.

- Mitlieferung des Lizenztextes
- Änderungshinweis bei Quellcode-Vertrieb

  Geänderte Quellcodedateien müssen mit einem Hinweis darauf versehen werden, dass diese modifiziert wurden (z. B. „modified by DLR"). Wird der geänderte Code über das Versionskontrollsystem Git an das Open-Source-Projekt übermittelt, bedarf es keines gesonderten Vermerkes, dass die entsprechenden Informationen in der History einsehbar sind. In allen anderen Fällen ist der Änderungsvermerk im Header aufzunehmen oder im Patch.

4.11 Apache License, version 2.0 (Apache-2.0)

4.11.1 Überblick

Die Apache Foundation hat die Apache-2.0 im Jahr 2004 eingeführt und damit die Apache-1.0 und Apache-1.1 abgelöst. Es handelt sich um eine Permissive Lizenz mit der Besonderheit, dass dem Lizenztext, anders als bei anderen Lizenzen ohne Copyleft, kein Copyright-Vermerk vorangestellt wird, sodass auch bei einer Mehrzahl von Rechteinhaberinnen bzw. Rechteinhabern nur ein Lizenztext mitzuliefern ist. Zudem enthält sie eine Klausel zum Umfang der Patentlizenzierung (sofern existierend).

4.11.2 Lizenzpflichten

Lizenzpflichten greifen erst bei einer Weiterverbreitung („distribution“) von Kopien der Software ein. Die rein interne Nutzung, d. h. innerhalb einer juristischen Person wie dem DLR, führt zu keinen Lizenzpflichten.

- Mitlieferung des Lizenztextes
- Beibehaltung von bereits bestehenden Marken-, Patent- und Urhebervermerken nur bei Quellcodevertrieb
- Änderungshinweis im Quellcode
- Mitlieferung der NOTICE-Datei

  Üblicherweise werden in der „NOTICE“-Datei Copyright-Vermerke und Lizenzhinweise gesammelt und somit gesammelt weitergegeben. Sofern eine solche Datei in dem Originalprojekt nicht existiert, muss sie auch nicht erstellt werden.

print("Empowerment of individ-uals is a key part of what makes open source work, since in the end, innovations tend to come from small groups, not from large, structured efforts.")

Tim O'Reilly

5 Praktische Anwendungsfälle beim Einsatz von Open-Source-Software

Nachfolgend werden einige Anwendungsfälle behandelt, die bei der Verwendung von Open-Source-Software relevant werden können:

- Vertrieb von eigenen Entwicklungen[32] ohne Open-Source-Software an Dritte
- Vertrieb von eigenen Entwicklungen mit Open-Source-Software an Dritte
- Angebot von eigenen Entwicklungen mit Open-Source-Software im Wege des SaaS
- Beiträge Dritter zu eigenen Entwicklungen
- Beiträge zu Open-Source-Projekten Dritter

Es ist zu beachten, dass die gewählte Veröffentlichungsform (siehe 3.2.2) die entstehenden Pflichten beeinflusst.

[32] Der Begriff „Entwicklung" ist ein Überbegriff für Anwendungen, Bibliotheken und sonstige Software-Programmierungen. In Kapitel 5 ist stets all dies gemeint, sodass einheitlich der Überbegriff „Entwicklung" gewählt wurde, außer im Zusammenhang mit der LGPL. Dort wird allein die Verknüpfung von eigenen Anwendungen mit fremden Bibliotheken behandelt, nicht aber die Verknüpfung von eigenen Bibliotheken mit fremden Bibliotheken.

5.1 Vertrieb von eigenen Entwicklungen ohne Open-Source-Software

Sofern Open-Source-Software nur bei der Programmierung eingesetzt wird, etwa durch Open-Source-Tools wie den GNU C Compiler oder das Eclipse Framework, dann sind die Open-Source-Lizenzen der Tools irrelevant für die damit erstellten eigenen Entwicklungen, solange sich kein Programmcode aus diesen Tools in den Eigenentwicklungen befindet. Die eigenen Entwicklungen können dann frei wählbar lizenziert werden, Lizenzpflichten aus den Open-Source-Lizenzen sind nicht zu erfüllen.

Sofern die Tools Code einkopieren oder vorbestehende Bibliotheken solcher Tools Teil der eigenen Entwicklungen werden, ist zunächst zu prüfen, ob das Open-Source-Softwaretool eine Lizenzausnahme vorsieht. Ist dies nicht der Fall, liegt der nachfolgend beschriebene Anwendungsfall „Vertrieb von eigenen Entwicklungen mit Open-Source-Software an Dritte“ vor.

5.2 Vertrieb von eigenen Entwicklungen und unveränderten Open-Source-Komponenten

Nahezu alle Open-Source-Lizenzen knüpfen die Erfüllung von Lizenzpflichten an eine Weiterverbreitung („distribution“) der Software. Eine rein interne Nutzung (produktiv oder zu Testzwecken) führt demnach nicht dazu, dass Lizenzpflichten zu erfüllen sind.

Eine Verbreitung liegt immer dann vor, wenn eine Kopie der Software an einen Dritten weitergegeben wird, sei es eine Kopie auf einem physikalischen Datenträger, sei es das Angebot zum Download, das einen unkörperlichen Vertrieb ermöglicht.

Eine Verbreitung im juristischen Sinne kann allerdings nicht nur dann vorliegen, wenn die physikalische Handlung der Verbreitung selbst vorgenommen wurde, sondern auch dann, wenn Dritte in den Gesamtvorgang einbezogen wurden. Hier kommt es darauf an, wer die organisatorische Kontrolle über den Vorgang hat. Bei dem Vertrieb eines Dockerfiles ist z. B. der Verbreitende des Dockerfiles auch für die dort referenzierte Software verantwortlich und nicht nur der Betreibende des Downloadangebots (wie z. B. Docker Hub). Ausgenommen sind nur Systemvoraussetzungen, wie z. B. Hinweise auf das zu verwendende Betriebssystem, auf dem eine Anwendung in-stalliert werden kann.

Ein Sonderfall liegt vor, wenn wie beim SaaS oder einem Cloud-Service zwar keine Kopie überlassen wird, der Kunde aber die Funktionalität der Software über ein Netzwerk benutzen kann. Hier hängt es von der anwendbaren Lizenz ab, ob Lizenzpflichten zu erfüllen sind. Darauf wird nachfolgend in den entsprechenden Anwendungsfällen hingewiesen.

Während die jeweiligen Vertriebspflichten aus den Open-Source-Lizenzen meist unabhängig davon zu erfüllen sind, ob die Software modifiziert wurde (z. B. Mitlieferung von Lizenztexten und Copyright-Vermerken), hängt die Anwendung eines Copyleft (siehe 3.1.3) einerseits von der Form der technischen Einbindung und andererseits von der Art des Copyleft in den anwendbaren Open-Source-Lizenzen ab. Entsprechend wird in den nachfolgend beschriebenen Anwendungsfällen danach unterschieden, ob Open-Source-Komponenten modifiziert werden und welche Lizenz einschlägig ist.

5.2.1 Vertrieb von eigenen Anwendungen, verlinkt mit unveränderten LGPL-2.1-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte LGPL-2.1-Bibliotheken, die unverändert mit

eigenen Anwendungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Das Copyleft der LGPL-2.1 bleibt auf die LGPL-2.1-Bibliothek beschränkt, wenn zum einen die Modifikation der eigenen Anwendung sowie das Reverse Engineering zum Zwecke der Fehlerbehebung ausdrücklich gestattet wird und zum anderen die Neuverlinkung für den Nutzenden möglich ist, wenn dieser die LGPL-2.1-Bibliothek später selbst modifizieren und mit der eigenen Anwendung neu verlinken möchte. Bei einer dynamischen Verlinkung reicht es aus, wenn ein sog. „suitable shared library mechanism" vorliegt, d. h. die Neuverlinkung bei einer einfachen Ersetzung einer (schnittstellenkompatiblen) Bibliothek funktioniert. Bei einer statischen Verlinkung muss dem Nutzenden auf Anfrage eine Objektcode-Datei der eigenen Anwendung zur Verfügung gestellt werden, sodass der Nutzende sich selbst ein Executable erstellen kann, wenn er sich eine modifizierte LGPL-2.1-Bibliothek selbst kompiliert und dann mit der eigenen Anwendung verlinkt.

Zu beachten ist, dass die LGPL-2.1 die Zulässigkeit von digitalen Signaturen (z. B. bei der Verwendung in sicherheitskritischen Embedded-Systemen) nicht explizit regelt. Da eine Neuverlinkung mit modifizierten LGPL-2.1-Bibliotheken stets verlangt, dass der Nutzende auch ein selbst erstelltes Executable oder die dynamisch zu verlinkende Bibliothek wieder auf das Embedded-System aufspielen kann, wird man zumindest verlangen müssen, dass dem Nutzenden die entsprechende Software auf Wunsch digital signiert wird oder die Abfrage so verändert wird, dass der Nutzende die Software auch mit seinem eigenen Schlüssel signieren kann.

Wenn die oben genannten Voraussetzungen erfüllt werden, muss die eigene Anwendung weder unter der LGPL-2.1 lizenziert noch im Quellcode zur Verfügung gestellt werden (Abb. 7).

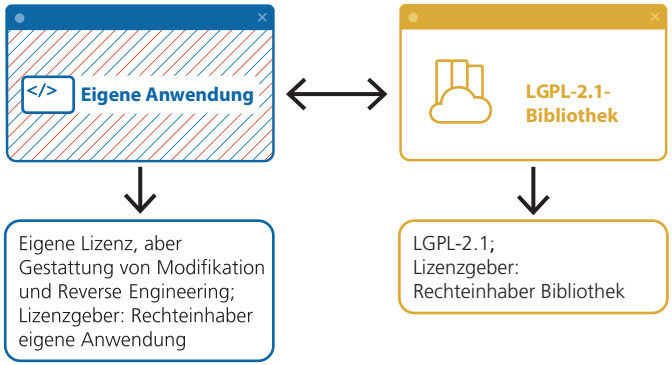


Abb. 7: Vertrieb von eigenen Anwendungen verlinkt mit unveränderten LGPL-2.1-Bibliotheken

5.2.2 Vertrieb von eigenen Anwendungen, verlinkt mit unveränderten LGPL-3.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte LGPL-3.0-Bibliotheken, die unverändert mit eigenen Anwendungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Das Copyleft ist wie bei der LGPL-2.1 geregelt und bleibt auf die LGPL-3.0-Bibliothek beschränkt. Allerdings darf die Bearbeitung der Bibliothek (als Teil der Kombination von Anwendung und Bibliothek) auch nicht beschränkt werden. Zudem muss das Reverse Engineering zur Fehlerbehebung solcher Bearbeitungen zulässig sein.

Die LGPL-3.0 verlangt grundsätzlich, dass der Nutzende die LGPL-3.0-Bibliothek durch eine modifizierte Version ersetzen können soll. Dem würde die Verwendung von digitalen Signaturen in Embedded-Systemen entgegenstehen. Die LGPL-3.0 sieht hier vor, dass bei Verbraucherprodukten das Wiederaufspielen und die Neuverlinkung von bearbeiteten Bibliotheken ermöglicht werden muss. Bei reinen Business-to-Business-Produkten ist dies nicht der Fall.

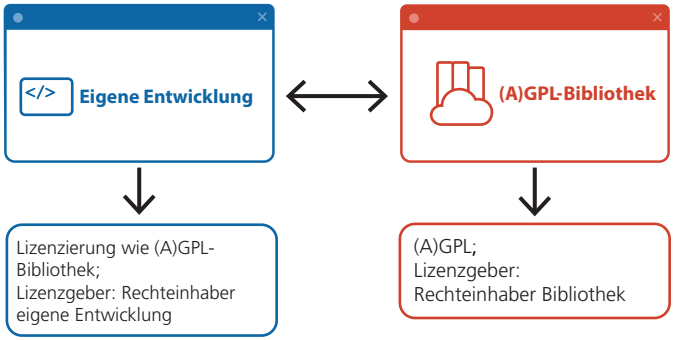


Abb. 8: Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten GPL-2.0/GPL-3.0/AGPL-3.0-Bibliotheken

5.2.3 Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten GPL-2.0/GPL-3.0/ AGPL-3.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte (A)GPL-Bibliotheken, die unverändert mit eigenen Entwicklungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Bei strengen Copyleft-Lizenzen wie der GPL-2.0, GPL-3.0 und AGPL-3.0 müssen verlinkte Komponenten ebenfalls unter der entsprechenden Copyleft-Lizenz lizenziert werden. Dies gilt übrigens unabhängig davon, ob eine (A)GPL-Bibliothek in eine eigene Entwicklung gelinkt oder eine eigenentwickelte Bibliothek mit einer (A)GPL-Anwendung verlinkt wird (Abb. 8).

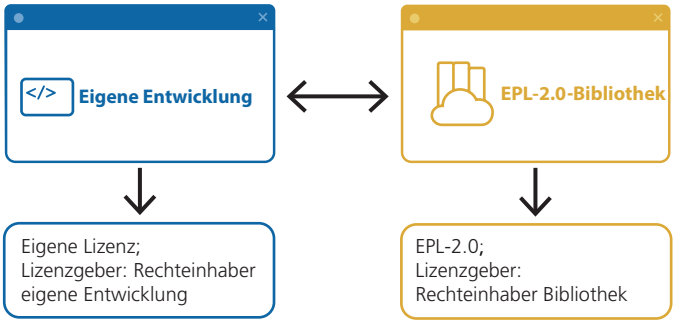


Abb. 9: Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten EPL-2.0-Bibliotheken

5.2.4 Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten EPL-2.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte EPL-2.0-Bibliotheken, die unverändert mit eigenen Entwicklungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Die EPL-2.0 findet vor allem im Eclipse-Umfeld und bei der Programmierung in Java Anwendung. Die EPL-2.0 beschränkt das Copyleft auf die jeweilige Klassenbibliothek, verlangt aber nicht, dass das gesamte Programm der EPL-2.0 unterstellt wird. Da die EPL-1.0 hierzu nicht eindeutig war, empfiehlt sich die Nutzung von EPL-1.0-lizenzierter Software unter der EPL-2.0, was die EPL-1.0 explizit gestattet (Abb. 9).

5.2.5 Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten MPL-2.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte MPL-2.0-Bibliotheken, die unverändert mit eigenen Entwicklungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Die MPL-2.0 enthält ein „file-based" Copyleft, dass auf Dateien beschränkt ist, die MPL-2.0-Code enthalten (Abb. 10).

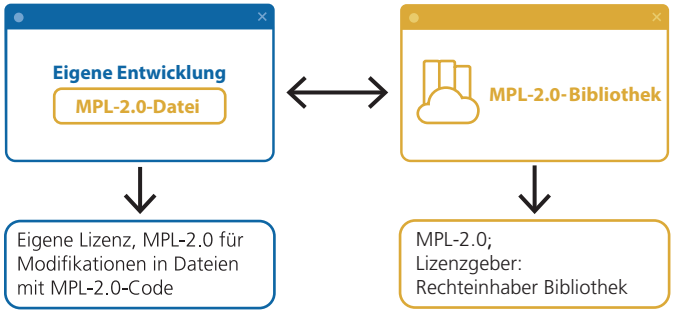


Abb. 10: Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten MPL-2.0-Bibliotheken

5.2.6 Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten Bibliotheken unter Permissiven Lizenzen (MIT, BSD-2-Clause, BSD-3-Clause, Apache-2.0 und Zlib)

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte Bibliotheken unter Permissiven Lizenzen, die unverändert mit eigenen Entwicklungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Permissive Lizenzen haben kein Copyleft und entsprechend können eigene Entwicklungen mit beliebigen Bibliotheken unter diesen Lizenzen verlinkt werden, ohne dass es Einschränkungen für die Lizenzwahl für die eigenen Entwicklungen gibt. Auch eine proprietäre Lizenzierung ist möglich. Die meisten Permissiven Lizenzen verlangen aber die Mitlieferung von Lizenztext und Copyright-Vermerken. Entsprechend sollte stets eine Open-Source-Lizenzinformation (d. h., ein Dokument oder eine Datei, in der alle zur Erfüllung der Lizenzpflichten mitzuliefernden Artefakte gesammelt sind) mitgeliefert werden
(Abb. 11).

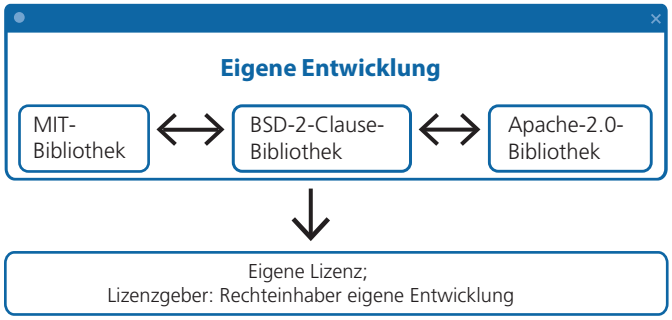


Abb. 11: Vertrieb von eigenen Entwicklungen, verlinkt mit unveränderten Bibliotheken unter Permissiver Lizenz (MIT, BSD-2-Clause, BSD-3-Clause, Apache-2.0 und Zlib)

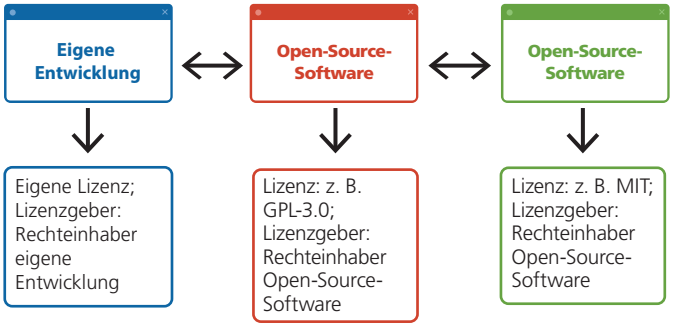


Abb. 12: Vertrieb von eigenen Entwicklungen, die mit Open-Source-Software über Schnittstellen Daten austauschen

5.2.7 Vertrieb von eigenen Entwicklungen, die mit Open Source Software über Schnittstellen Daten austauschen

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen und Open-Source-Software zusammen an Dritte weitergegeben werden, aber nicht in ein Programm integriert wurden, sondern unabhängig nebeneinanderstehen und allenfalls über vorbestehende Kommunikationsschnittstellen Daten austauschen (z. B. über Sockets mittels TCP/IP oder Pipes).

Die bloße Kommunikation im Sinne eines Datenaustauschs durch eine definierte Schnittstelle führt nach ganz überwiegender Auffassung nicht zu einer Bearbeitung eines der betroffenen Programme und daher auch nicht zu Copyleft-Effekten. Daher dürfen auf diese Weise Open-Source-Programme unter beliebigen Lizenzen untereinander und mit eigenen Entwicklungen zusammen vertrieben werden. Sämtliche Lizenzpflichten der beteiligten Open-Source-Software sind zu erfüllen (Abb. 12).

Unklar ist, wie die Rechtslage bei anderen Schnittstellen als reinen Kommunikationsschnittstellen ist. Daher sind im Zweifel die Anforderungen wie bei einer Verlinkung zu erfüllen.

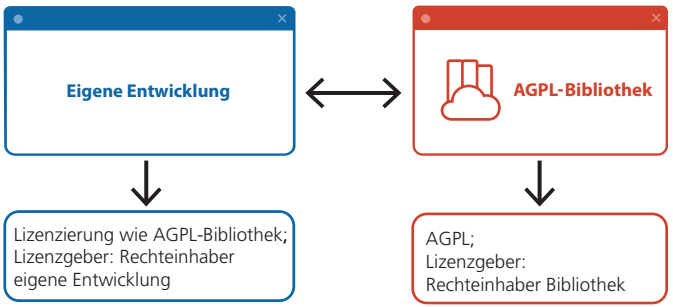


Abb. 13: Angebot von eigenen Entwicklungen, die unveränderte Bibliotheken unter der AGPL-3.0 verwenden, im Wege des SaaS

5.2.8 Angebot von eigenen Entwicklungen, die unveränderte Bibliotheken unter der AGPL-3.0 verwenden, im Wege des SaaS

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen und AGPL-3.0-Bibliotheken zusammen im Wege des SaaS zur Nutzung angeboten werden.

Ziffer 13 AGPL-3.0 sieht Lizenzpflichten bei einer Nutzung im Wege des SaaS nur dann vor, wenn die AGPL-3.0-Software modifiziert wurde. Da eine Verlinkung bei der AGPL-3.0 bereits

als „Modification“ im Sinne der Lizenz gesehen wird, muss der Quellcode der eigenen Anwendung auch unter den Lizenzbedingungen der AGPL-3.0 zur Verfügung gestellt werden. Bei der Verwendung von AGPL-3.0-Snippets in einer eigenen Entwicklung gilt dies entsprechend (Abb. 13).

5.2.9 Angebot von eigenen Entwicklungen, die unveränderte Bibliotheken unter der GPL-3.0, LGPL-3.0, EPL-2.0, MPL-2.0 oder Apache-2.0 verwenden, im Wege des SaaS

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen oder kurze Codeabschnitte (Snippets) unter der GPL-3.0, LGPL-3.0, EPL-2.0, MPL-2.0 oder Apache-2.0 zusammen im Wege des SaaS zur Nutzung angeboten werden.

Die genannten Lizenzen sind, bis auf die Apache-2.0, Copyleft-Lizenzen, knüpfen aber Lizenzpflichten nur an den Vertrieb von Kopien an Dritte. Das wäre zum Beispiel der Fall, wenn Javascript an den Browser des Nutzenden übertragen wird. Bei einem reinen SaaS ohne Übertragung von Programmcode sind keinerlei Lizenzpflichten zu erfüllen und die eigene Entwicklung kann proprietär vermarktet werden.

5.2.10 Angebot von eigenen Entwicklungen, die unveränderte Bibliotheken unter der GPL-2.0 oder LGPL-2.1 verwenden, im Wege des SaaS

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen oder kurze Codeabschnitte unter der GPL-2.0 oder LGPL-2.1 zusammen im Wege des SaaS zur Nutzung angeboten werden.

Unklar und umstritten ist, ob ein SaaS von GPL-2.0 oder LGPL-2.1-Programmen unter den Begriff der „distribution“ fällt und Lizenzpflichten nach sich zieht. Sofern eine interaktive Nutzung mit GPL-2.0- oder LGPL-2.1-Software stattfindet, sollten im Zweifel die Lizenzpflichten erfüllt werden, sodass auf die Anwendungsfälle unter 5.2.1 und 5.2.3 verwiesen werden kann. Wenn jedoch keine interaktive Nutzung stattfindet, z. B. bei einem Linux-Betriebssystem unter der GPL-2.0 zum Betrieb des Servers, liegt eine rein interne Nutzung vor, die keine Lizenzpflichten nach sich zieht.

5.3 Vertrieb von eigenen Entwicklungen und veränderten Open-Source-Komponenten

Werden Open-Source-Komponenten verändert, stellen sich zwei Fragen im Hinblick auf die Auswirkungen von Copyleft-Klauseln: (1) Welche Pflichten ergeben sich im Hinblick auf die veränderte Open-Source-Komponente? und (2) Welche Auswirkungen hat das Copyleft für die eigene Entwicklung?

5.3.1 Vertrieb von eigenen Anwendungen, verlinkt mit veränderten LGPL-2.1-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte LGPL-2.1-Bibliotheken, die in veränderter Form mit eigenen Anwendungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Das Copyleft der LGPL-2.1 bleibt auf die LGPL-2.1-Bibliothek beschränkt, wenn zum einen die Modifikation der eigenen Anwendung sowie das Reverse Engineering zum Zwecke der Fehlerbehebung ausdrücklich gestattet werden und zum anderen die Neuverlinkung für den Nutzenden möglich ist, wenn dieser die LGPL-2.1-Bibliothek später selbst modifizieren und mit der eigenen Anwendung neu verlinken möchte (siehe 5.2.1). Der Quellcode der veränderten LGPL-2.1 muss mitgeliefert oder den Kundinnen bzw. Kunden angeboten werden.

Zur Verwendung von digitalen Signaturen siehe 5.2.1.

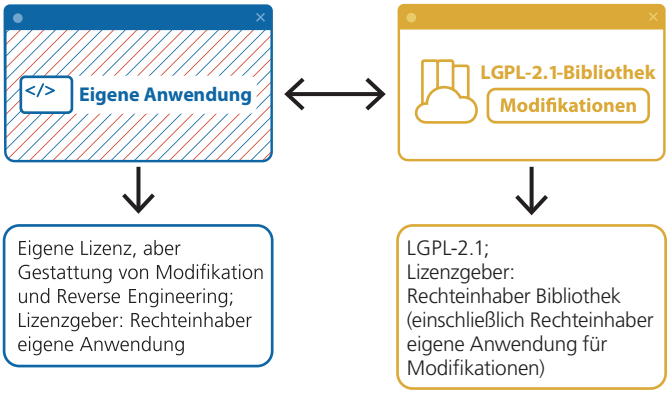


Abb. 14: Vertrieb von eigenen Anwendungen, verlinkt mit veränderten LGPL-2.1-Bibliotheken

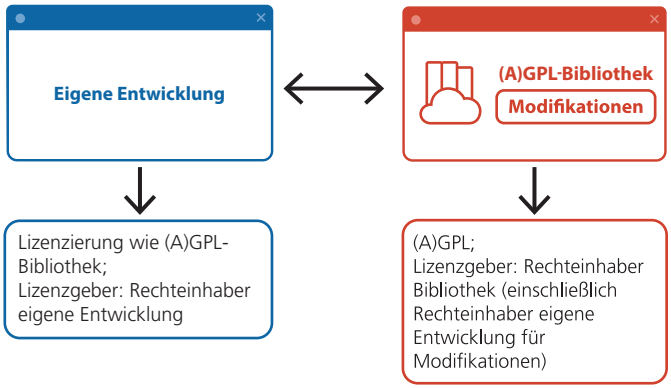


Abb. 15: Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten GPL-2.0/GPL-3.0/AGPL-3.0-Bibliotheken

Wenn die oben genannten Voraussetzungen erfüllt werden, muss die eigene Entwicklung weder unter der LGPL-2.1 lizenziert noch im Quellcode zur Verfügung gestellt werden. Anders ist die Lage nur, wenn Code unter der LGPL-2.1 in die eigene Entwicklung einkopiert wird, bei Code aus einem Headerfile nur dann, wenn es sich um mehr als 10 Zeilen Code handelt und diese nicht nur aus numerischen Parametern, Datenstrukturanordnungen und Zugriffsfunktionen bestehen (Abb. 14).

5.3.2 Vertrieb von eigenen Anwendungen, verlinkt mit veränderten LGPL-3.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte LGPL-3.0-Bibliotheken, die in veränderter Form mit eigenen Anwendungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Das Copyleft ist wie bei der LGPL-2.1 geregelt und bleibt auf die LGPL-Bibliothek beschränkt (siehe 5.3.1). Allerdings darf die Bearbeitung der Bibliothek (als Teil der Kombination von Anwendung und Bibliothek) auch nicht beschränkt werden. Zudem muss das Reverse Engineering zur Fehlerbehebung solcher Bearbeitungen zulässig sein.

Zur Verwendung von digitalen Signaturen siehe 5.2.1.

5.3.3 Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten GPL-2.0/GPL-3.0/ AGPL-3.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte (A)GPL-Bibliotheken, die in veränderter Form mit eigenen Entwicklungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Bei strengen Copyleft-Lizenzen wie der GPL-2.0, GPL-3.0 und AGPL-3.0 müssen sowohl die Modifikationen der GPL-2.0/GPL-3.0/AGPL-3.0-Bibliotheken als auch die damit verlinkten Komponenten unter der entsprechenden Copyleft-Lizenz lizenziert und im Quellcode zur Verfügung gestellt werden. Dies gilt übrigens unabhängig davon, ob eine (A)GPL-Bibliothek in eine eigene Entwicklung gelinkt oder eine eigenentwickelte Bibliothek mit einer (A)GPL-Anwendung verlinkt wird (Abb. 15).

5.3.4 Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten EPL-2.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte EPL-2.0-Bibliotheken, die in veränderter Form mit eigenen Entwicklungen oder Bibliotheken an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Die EPL-2.0 findet vor allem im Eclipse-Umfeld und bei der Programmierung in Java Anwendung. Die EPL-2.0 beschränkt das Copyleft auf die jeweilige Klassenbibliothek, verlangt aber nicht, dass das gesamte Programm der EPL-2.0 unterstellt werden muss. Da die EPL-1.0 hierzu nicht eindeutig war, empfiehlt sich die Nutzung von EPL-1.0-lizenzierter Software unter der EPL-2.0, was die EPL-1.0 explizit gestattet (Abb. 16).

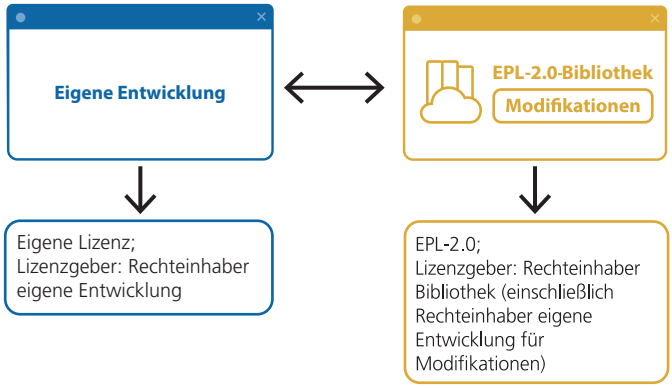


Abb. 16: Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten EPL-2.0-Bibliotheken

5.3.5 Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten MPL-2.0-Bibliotheken

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte MPL-2.0-Bibliotheken, die in veränderter Form mit eigenen Anwendungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Die MPL-2.0 enthält ein „file-based" Copyleft, das auf Dateien beschränkt ist, die MPL-2.0-Code enthalten. Somit ist sowohl bei der MPL-2.0-Bibliothek als auch im Hinblick auf die eigene

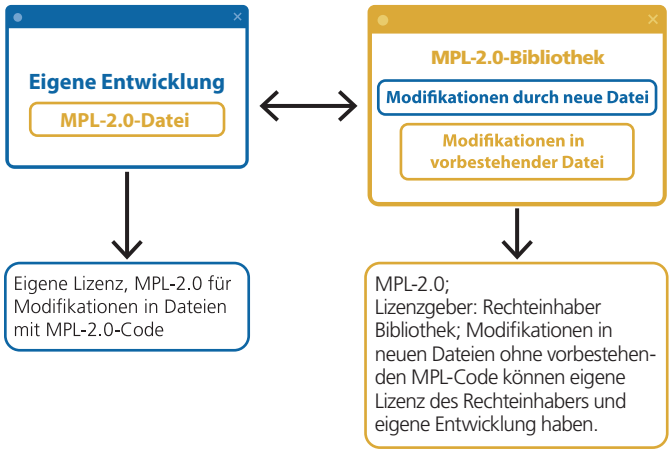


Abb. 17: Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten MPL-2.0-Bibliotheken

Entwicklung zu prüfen, ob die jeweilige Datei vorbestehenden MPL-2.0-Code enthält und dann unter der MPL-2.0 lizenziert werden muss (Abb. 17)

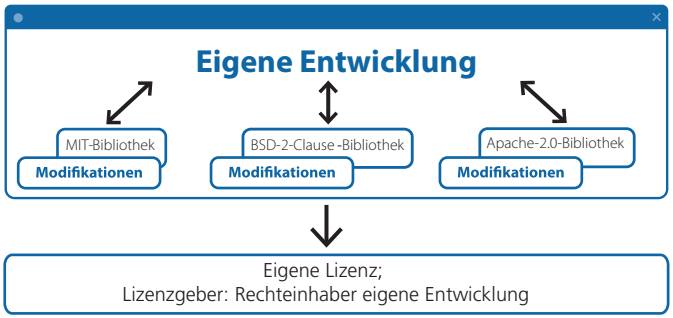


Abb. 18: Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten Bibliotheken unter Permissiver Lizenz (MIT, DSD-2-Clause, BSD-3-Clause, Apache-2.0 und Zlib)

5.3.6 Vertrieb von eigenen Entwicklungen, verlinkt mit veränderten Bibliotheken unter Permissiven Lizenzen (MIT, BSD-2-Clause, BSD-3-Clause, Apache-2.0 und Zlib)

Dieser Anwendungsfall erfasst sowohl statisch als auch dynamisch verlinkte Bibliotheken unter Permissiven Lizenzen, die modifiziert mit eigenen Anwendungen an Dritte weitergegeben werden, sei es per Download oder auf einem Datenträger.

Permissive Lizenzen haben kein Copyleft und entsprechend können eigene Entwicklungen sowohl mit beliebigen Bibliotheken unter diesen Lizenzen verlinkt werden als auch modifizierte Bibliotheken verwenden, ohne dass es Einschränkungen für die Lizenzwahl für die eigenen Entwicklungen gibt. Auch eine proprietäre Lizenzierung ist möglich. Die meisten Permissiven Lizenzen verlangen aber die Mitlieferung von Lizenztext und Copyright-Vermerken. Entsprechend sollte stets eine Open-Source-Lizenzinformation mitgeliefert werden (Abb. 18).

5.3.7 Vertrieb von eigenen Entwicklungen mit Code-Snippets unter Copyleft-Lizenzen

Dieser Anwendungsfall betrifft die Konstellation, dass nicht komplette Komponenten verwendet werden, sondern Code-Snippets (z. B. einzelne Funktionen) unter Copyleft-Lizenzen (z. B. AGPL-3.0, GPL-3.0, GPL-2.1, LGPL-3.0, LGPL-2.1, EPL-2.0, MPL-2.0), die in bestehende Quellcodedateien von eigenen Entwicklungen einkopiert werden.

Die Verwendung von Code-Snippets in eigenen Entwick-

lungen löst stets das Copyleft einer Copyleft-Lizenz aus. Während bei AGPL-3.0, GPL-3.0 und GPL-2.1 das gesamte Programm unter der jeweiligen Copyleft-Lizenz lizenziert werden muss, hängt der Umfang des Copyleft bei LGPL-3.0, LGPL-2.1 und EPL-2.0 davon ab, ob der Code-Snippet in eine Bibliothek einkopiert wurde – dann fällt nur diese unter das Copyleft, oder in eine Anwendung oder ein anderes Programm – dann fällt das gesamte Programm unter das Copyleft. Eine Ausnahme ist die MPL-2.0 mit ihrem „file-based" Copyleft, das auf Dateien beschränkt ist, die MPL-2.0-Snippets enthalten.

5.3.8 Vertrieb von eigenen Entwicklungen mit Code-Snippets unter Permissiven Lizenzen

Dieser Anwendungsfall betrifft die Konstellation, dass nicht komplette Komponenten verwendet werden, sondern Code-Snippets (z. B. einzelne Funktionen) unter Permissiven Lizenzen (z. B. MIT, BSD-2-Clause, BSD-3-Clause, Apache-2.0 und Zlib), die in bestehende Quellcodedateien von eigenen Entwicklungen einkopiert werden.

Mangels Copyleft ergibt sich bei der Verwendung von Snippets keine Abweichung zur Verwendung ganzer Bibliotheken (siehe 5.2.6). Allerdings sollte bei einem Vertrieb im Quellcode (z. B. bei Skript-Sprachen) darauf geachtet werden, dass die Header der Dateien mit Snippet-Code deutlich machen, dass die Datei zwar insgesamt unter der gewählten Lizenz für das gesamte Programm lizenziert ist, aber auch Bestandteile unter einer Permissiven Lizenz enthält. Dabei sollten auch der Lizenztext und ggf. Copyright-Vermerke angegeben werden.

5.3.9 Vertrieb von eigenen Entwicklungen, die mit veränderter Open-Source-Software über Schnittstellen Daten austauschen

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen und Open-Source-Software zusammen an Dritte weitergegeben werden, aber nicht in ein Programm integriert wurden, sondern unabhängig nebeneinanderstehen und allenfalls über vorbestehende Kommunikationsschnittstellen Daten austauschen (z. B. über Sockets mittels TCP/IP oder Pipes).

Entsprechend den Hinweisen unter 5.2.7 kann die eigene Entwicklung unter beliebigen Lizenzbedingungen stehen, da es sich im Regelfall um ein unabhängiges Werk handelt. Hinsichtlich der modifizierten Open-Source-Software hängen die Auswirkungen eines Copyleft von der betroffenen Lizenz ab (siehe 5.3.1 – 5.3.8). Sollte die Bearbeitung einer Open-Source-Software erforderlich sein, um überhaupt eine Kommunikation mit der eigenen Entwicklung herstellen zu können, empfiehlt sich eine Einzelfallprüfung, wenn die Open-Source-Software unter der GPL-2.0, GPL-3.0 oder AGPL-3.0 lizenziert ist (Abb. 19).

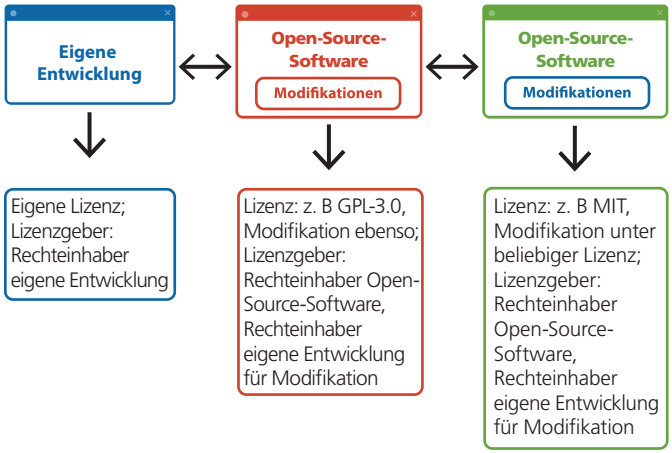


Abb. 19: Vertrieb von eigenen Entwicklungen, die mit veränderter Open-Source-Software über Schnittstellen Daten austauschen

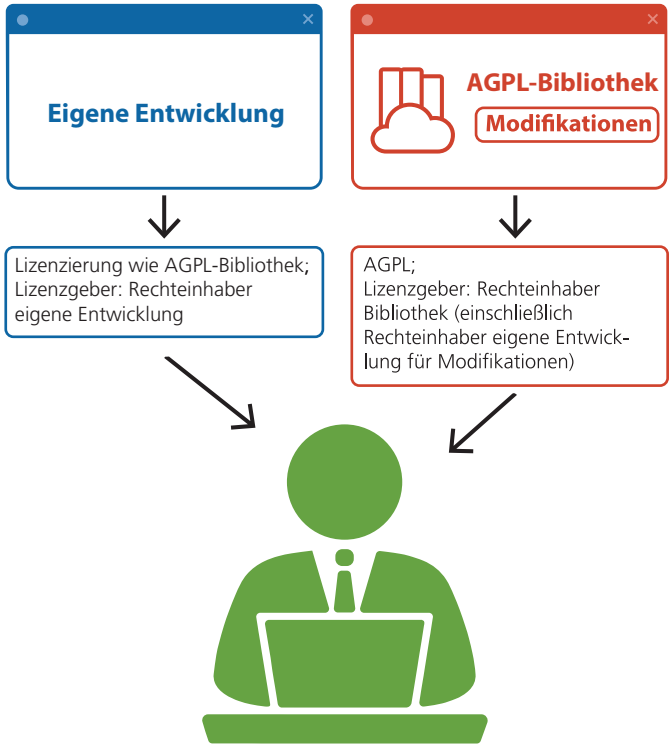


Abb. 20: Angebot von eigenen Entwicklungen, die veränderte Bibliotheken oder Funktionalitäten unter der AGPL-3.0 verwenden, im Wege des SaaS

5.3.10 Angebot von eigenen Entwicklungen, die veränderte Bibliotheken oder Funktionalitäten unter der AGPL-3.0 verwenden, im Wege des SaaS

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen und AGPL-3.0-Bibliotheken oder AGPL-Snippets zusammen im Wege des SaaS zur Nutzung angeboten werden. Wenn der Nutzende die Funktionalität der veränderten AGPL-3.0-Software interaktiv nutzen kann („interacting with it remotely through a computer network"), muss der Quellcode der gesamten Software, die die AGPL-3.0-Bestandteile oder Bibliotheken enthält, unter den Lizenzbedingungen der AGPL-3.0 per Downloadmöglichkeit zur Verfügung gestellt werden (Abb. 20).

5.3.11 Angebot von eigenen Entwicklungen, die veränderte Bibliotheken oder Funktionalitäten unter der GPL-3.0, LGPL-3.0, EPL-2.0, MPL-2.0 oder Apache-2.0 verwenden, im Wege des SaaS

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen und Bibliotheken oder Snippets unter der GPL-3.0, LGPL-3.0, EPL-2.0, MPL-2.0 oder Apache-2.0 zusammen im Wege des SaaS zur Nutzung angeboten werden.

Die genannten Lizenzen sind allesamt Lizenzen, die bei einem reinen SaaS ohne Übertragung von Programmcode keinerlei Lizenzpflichten vorsehen. Dies gilt sowohl hinsichtlich eines Copyleft als auch im Hinblick auf Vertriebspflichten, die nur bei einem Vertrieb einer Kopie zu erfüllen sind.

5.3.12 Angebot von eigenen Entwicklungen, die veränderte Bibliotheken oder Funktionalitäten unter der GPL-2.0 oder LGPL-2.1 verwenden, im Wege des SaaS

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen und Bibliotheken oder Snippets unter der GPL-2.0 oder LGPL-2.1 zusammen im Wege des SaaS zur Nutzung angeboten werden.

Wie bereits oben unter 5.2.10 ausgeführt, ist unklar und umstritten, ob ein SaaS von GPL-2.0- oder LGPL-2.1-Programmen unter den Begriff der „distribution" fällt und Lizenzpflichten nach sich zieht. Sofern eine interaktive Nutzung mit GPL-2.0- oder LGPL-2.1-Software stattfindet, sollten im Zweifel die Lizenzpflichten erfüllt werden, sodass auf die Anwendungsfälle unter 5.3.1 und 5.3.3 verwiesen werden kann. Wenn jedoch keine interaktive Nutzung stattfindet, z. B. bei einem Linux-Betriebssystem unter der GPL-2.0 zum Betrieb des Servers, liegt eine rein interne Nutzung vor, die keine Lizenzpflichten nach sich zieht.

5.4 Beiträge Dritter zu eigenen Entwicklungen

Dieser Anwendungsfall erfasst die Situation, dass Dritte zu eigenen Entwicklungen beitragen. Die Beiträge Dritter müssen analog zu eigenen Beiträgen überprüft werden. Dies bedeutet, dass die gleichen Regelungen gelten wie für die Anwendungsfälle unter 5.1 bis 5.3.

Mittels eines Developer Certificate of Origins (DCO) kann von Beitragenden eine Zusicherung verlangt werden, dass es sich um einen rechtlich zulässigen Beitrag handelt (siehe 3.2.3). Weiterhin wird der Beitragende Bearbeiterurheber und erhält dementsprechende Rechte. Diese können vertraglich (z. B. mittels eines CLA) übertragen werden. Wie zuvor unter 3.2.3 erläutert, können CLAs aber eine abschreckende Wirkung haben und sollten nur eingesetzt werden, wenn eine Rechteübertragung an den Rechteinhaber der eigenen Entwicklungen zwingend nötig ist. Das Recht auf Namensnennung ist davon nicht betroffen. Der Beitragende muss daher in die Autorenliste aufgenommen werden.

5.5 Beiträge zu Open-Source-Software Dritter

Dieser Anwendungsfall erfasst die Situation, dass eigene Entwicklungen zu Open-Source-Software Dritter beigetragen werden. Die Gründe, zu einem Open-Source-Projekt Dritter beizutragen, sind vielseitig. Es kann sich um eine gemeinsame Entwicklung handeln, die Ergänzung einer benötigten Funktionalität oder auch nur das Beheben eines kleinen Fehlers. Unterschieden wird vor allem nach der Form, in welcher ein Beitrag erfolgt.

5.5.1 Direkter Beitrag zu einem Open-Source-Projekt Dritter

Beiträge zu Open-Source-Projekten sind zwar in der Regel wenig umfangreich, stellen aber trotzdem Veröffentlichungen dar und unterliegen damit den gleichen rechtlichen und organisatorischen Regelungen wie unter 2.2 sowie in 5.1 bis 5.3 beschrieben.

Der damit einhergehende Aufwand reduziert sich in der Regel aber erheblich. So ist die Lizenz bereits festgelegt und oft kann auch von einer exportkontrollrechtlichen Überprüfung

abgesehen werden, wenn der Funktionsumfang der Software nicht verändert wird.[33] Enthält der Beitrag Code Dritter, muss jedoch weiterhin eine Lizenzprüfung erfolgen. Es empfiehlt sich also trotzdem, den unter 2.2 dargestellten Prozess durchzuarbeiten.

Folgende Besonderheiten gilt es zu beachten:

- Die Lizenz des Open-Source-Projektes sollte auf ihre Eignung bezüglich des Einsatzes in der eigenen Organisation überprüft werden. Dies ist insbesondere relevant, wenn regelmäßige oder umfangreiche Beiträge angestrebt werden.

- Wird regelmäßig zu einem Projekt beigetragen, sollte dies nicht jedes Mal vom Leitenden des Instituts bzw. der Einrichtung genehmigt werden müssen; hier sollte daher eine einmalige Abstimmung erfolgen.

- Wird das Unterschreiben eines CLAs oder anderer zusätzlicher Vertragsbedingungen gefordert, muss dies mit der Instituts- bzw. Einrichtungsleitung geklärt werden. Gegebenenfalls sollte eine Überprüfung unter Einbeziehung rechtlicher Beraterinnen und Berater erfolgen.

5.5.2 Eigenständige Erweiterung eines Open-Source-Projektes Dritter

Ein Fork ist eine Kopie einer Open-Source-Software, die unabhängig vom Ursprungsprojekt weiterentwickelt wird. Ein Fork kommt in der Regel zustande, weil ein Projekt Beiträge nicht akzeptiert oder die Entwickelnden die Weiterentwicklung einstellen. In diesem Fall handelt es sich nicht mehr um einen klassischen Beitrag, sondern um eine eigene Entwicklung auf Basis von Code Dritter. Dies entspricht in der Regel dem Anwendungsfall „Vertrieb von eigenen Entwicklungen und veränderten Open-Source-Komponenten“ (siehe 5.3). In diesem Fall gelten die rechtlichen und organisatorischen Regelungen aus 2.2, unter Berücksichtigung der Ausgangssituation, dass bereits Urheber existieren.

[33] Von einer exportkontrollrechtlichen Überprüfung kann hier dann abgesehen werden, wenn die Software vorher bereits allgemein zugänglich war und durch die Zuarbeit nicht wesentlich geändert wird. Ausgenommen ist Software für Wehrtechnik und Trägertechnologien. Im Zweifel sprechen Sie immer die zuständige Exportkontrollabteilung in Ihrer Organisation an.

DLR-Technologietransfer

Im Vorstandsressort „Innovation, Transfer und wissenschaftliche Infrastrukturen" unterstützt der Bereich Technologietransfer die Institute und Einrichtungen des DLR dabei, Ergebnisse aus Forschung und Entwicklung in die Wirtschaft zu bringen und ist der zentrale Kontakt des DLR für gewerbliche Schutzrechte, Lizenzverträge und die damit zusammenhängenden Fragen.

Durchschnittlich bringt das DLR jährlich ca. 200 Erfindungen und entsprechende Schutzrechtsanmeldungen hervor. Dies führt zu einem regelmäßigen Bestand von ca. 3.800 Schutzrechten in mehr als 1.000 Patentfamilien. Mit der Anmeldung von Erfindungen kommt das DLR einerseits seiner gesellschaftspolitischen Aufgabe der Veröffentlichung seiner Forschungsergebnisse nach, zum anderen werden dadurch die Forschungsfelder des DLR abgesichert und Wettbewerbsvorteile für die Vermarktungspartner des DLR geschaffen. Im Rahmen von Vermarktungskooperationen begleiten wir unsere Institute und Vermarktungspartner bei der Ausarbeitung von zur jeweiligen Situation passenden Vereinbarungen, insbesondere Lizenzverträgen.

DLR-Institut für Softwaretechnologie

Die Aufgaben des DLR-Instituts für Softwaretechnologie sind Forschung und Entwicklung auf dem Gebiet innovativer Software-Engineering-Technologien und die Bereitstellung und Anwendung dieses Software-Know-hows. Die derzeitigen Themenschwerpunkte sind die Softwareentwicklung für verteilte und intelligente Systeme, Künstliche Intelligenz, Software für eingebettete Systeme, Visualisierung und High Performance Computing.

Die Forschungsgruppe Sustainable Software Engineering der Abteilung Intelligente und Verteilte Systeme unterstützt neben ihrer Forschungstätigkeit Forschende praktisch und praxisnah bei der nachhaltigen Entwicklung von Forschungssoftware. Die Gruppe vermittelt jahrelange Erfahrungen im Software Engineering und stellt Kenntnisse im Bereich Open Source und Open Science Wissenschaftlerinnen und Wissenschaftlern zur Verfügung.

Das DLR im Überblick

Das DLR ist das Forschungszentrum der Bundesrepublik Deutschland für Luft- und Raumfahrt. Wir betreiben Forschung und Entwicklung in Luftfahrt, Raumfahrt, Energie und Verkehr, Sicherheit und Digitalisierung. Die Deutsche Raumfahrtagentur im DLR ist im Auftrag der Bundesregierung für die Planung und Umsetzung der deutschen Raumfahrtaktivitäten zuständig. Zwei DLR Projektträger betreuen Förderprogramme und unterstützen den Wissenstransfer. Global wandeln sich Klima, Mobilität und Technologie.

Das DLR nutzt das Know-how seiner 55 Institute und Einrichtungen, um Lösungen für diese Herausforderungen zu entwickeln. Unsere 10.000 Mitarbeitenden haben eine gemeinsame Mission: Wir erforschen Erde und Weltall und entwickeln Technologien für eine nachhaltige Zukunft. So tragen wir dazu bei, den Wissens- und Wirtschaftsstandort Deutschland zu stärken.

Impressum

Herausgeber:
Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)

Redaktion und Inhalte:
Mathias Brochhagen, Carina Haupt, Dr. Till Jaeger (JBB Rechtsanwälte), Verena Müller, Tobias Schlauch, Dr. Konstantin Stumvoll

Anschrift:
Linder Höhe, 51147 Köln
E-Mail: info-dlr@dlr.de
DLR.de

Gestaltung:
CD Werbeagentur GmbH

Bilder: Adobe Stock – Gorodenkoff Productions OU

Abdruck (auch von Teilen) oder sonstige Verwendung nur nach vorheriger Absprache mit dem DLR gestattet.

Gefördert durch:
Bundesministerium für Wirtschaft und Klimaschutz
aufgrund eines Beschlusses des Deutschen Bundestages

DLR Deutsches Zentrum für Luft- und Raumfahrt

Nutzung von Open-Source-Software im DLR_D_07/2022